Berichte des Forschungsschwerpunkts
Ambient Intelligence

Nr. 5

Entwurf eines Reglers für das AmI-Szenario Assisted Training

Ankang Le
Oliver Gabel

Inhaltsverzeichnis

1 Einleitung

1.1 Einleitung und Motivation

Im Rahmen des Szenarios *Assisted Training* im Forschungsschwerpunkt Ambient Intelligence (AmI) der Technische Universität Kaiserslautern ist ein AmI-System zu entwickeln [Web-04]. Die Aufgabe des AmI-Systems ist die Optimierung des Trainingseffektes für eine Rennradgruppe von 2 bis 30 Radfahrern. Dabei, ist zuerst ein Fahrrad-Demonstrator zu entwickeln. Der Fahrrad-Demonstrator umfasst vier Rennräder und hat die Aufgabe, die Differenzen zwischen aktuellem Training und Trainingsplan für alle Radfahrer gleichermaßen zu minimieren.

Auf der Suche nach einem einfachen Parameter zur Intensitätssteuerung beim Training hat sich die Herzfrequenz bewährt und weltweit in den Ausdauersportarten durchgesetzt. In der Herzfrequenz schlagen sich alle inneren und äußeren Belastungsfaktoren nieder. Hinzu kommt, dass die Herzfrequenz einfach zu messen ist.

Diese Masterarbeit gliedert sich in die Entwicklung des Fahrrad-Demonstrators ein und befasst sich nur mit einem Fahrrad. Ziel dieser Arbeit ist die Konzeptionierung und Realisierung einer Herzfrequenzregelung mit unterlagerter Leistungsregelung für den Fahrrad-Demonstrator. Das heißt, die Abweichung der aktuellen Herzfrequenz von der Zielfrequenz ist zu minimieren.

Die in dieser Arbeit zu realisierenden Aufgaben erstrecken sich über zwei Ebenen. Die Aufgaben der unteren Ebene befassen sich mit der Hardware. Dabei ist die Wirbelstrombremse des Ergotrainers zu identifizieren, die Erfassung und die Auswertung der Sensordaten auf einer Sensorplatine zu implementieren. Daraufhin sind alle Sensordaten an einen PC zu übertragen. Die obere Ebene befasst sich nur mit der Software. Hierbei ist das Mensch-Modell mit dem Verfahren der kleinsten Quadrate zu identifizieren und daraufhin der Herzfrequenzregler mit einem modellbasierten Verfahren zu entwerfen. Anschließend ist das gesamte Regelungssystem auf einem PC in MATLAB/SIMULINK zu implementieren.

1.2 Kapitelübersicht

Kapitel 2 behandelt die eigentliche Aufgabenstellung dieser Masterarbeit und die Beschreibung des zu realisierenden Regelungssystems im Rahmen des AmI-Szenarios *Assisted Training*.

Kapitel 3 beschäftigt sich mit der Beschreibung der in dieser Arbeit verwendeten Hardwarekomponenten: der Flow Ergotrainer der Firma Tacx, der Herzfrequenzsensor der AG Medion,

die im Rahmen von AmI entwickelte Sensorplatine und deren zentrale Einheit, der Mikroprozessor ATmega128L der Firma Atmel.

In Kapitel 4 wird ein Konzept des zu erstellenden Regelungssystems entwickelt. Dabei wird zuerst das Blockschaltbild des gesamten Regelungssystems erstellt. Weiterhin werden die Datenerfassungsverfahren aller Sensorsignale vorgestellt. Außerdem werden die im System beteiligten PC- und Benutzerschnittstellen erläutert.

Im Anschluss wird in Kapitel 5 der Herzfrequenzregler entworfen. Hierbei wurden viele Stufentests mit unterschiedlichen Testprobanden durchgeführt. Auf dieser Basis wird ein mathematisches Modell für das Mensch-Fahrrad-System identifiziert. Danach wird der Herzfrequenzregler auf Basis des identifizierten Mensch-Modells mit einem modellbasierten Verfahren entworfen.

In Kapitel 6 wird auf die verschiedenen Aspekte der Software, welche zum Betrieb des Fahrrad-Demonstrators notwendig sind, eingegangen. An dieser Stelle werden zu einem die Software für den Mikroprozessor und zum anderen die MATLAB/SIMULINK-Programme erklärt. Daraufhin wird das gesamte Regelungssystem in SIMULINK mit Anti-Windup-Maßnahme implementiert.

Kapitel 7 befasst sich dann mit der Inbetriebnahme des Regelungssystems. Dabei werden die Validierungsergebnisse an dem Fahrrad-Demonstrator und das Verhalten des Herzfrequenzreglers vorgestellt.

Zum Schluss fasst Kapitel 8 diese Arbeit zusammen und stellt Erweiterungsmöglichkeiten des erstellten Systems im Rahmen des AmI-Szenarios *Assisted Training* vor.

2 Aufgabenstellung und Systembeschreibung

2.1 Aufgabenstellung

In dieser Masterarbeit wird die Konzeption und Realisierung einer Herzfrequenzregelung mit unterlagerter Leistungsregelung für den Fahrrad-Demonstrator vorgestellt. Das System des Fahrrad-Demonstrators ist für das Fahrradszenario des Forschungsschwerpunktes AmI der Technische Universität Kaiserslautern entwickelt worden. Der Demonstrator umfasst vier Rennräder, die mit verschiedenen Sensoren und einem Funknetzwerk ausgerüstet sind (siehe Abbildung 2.1).

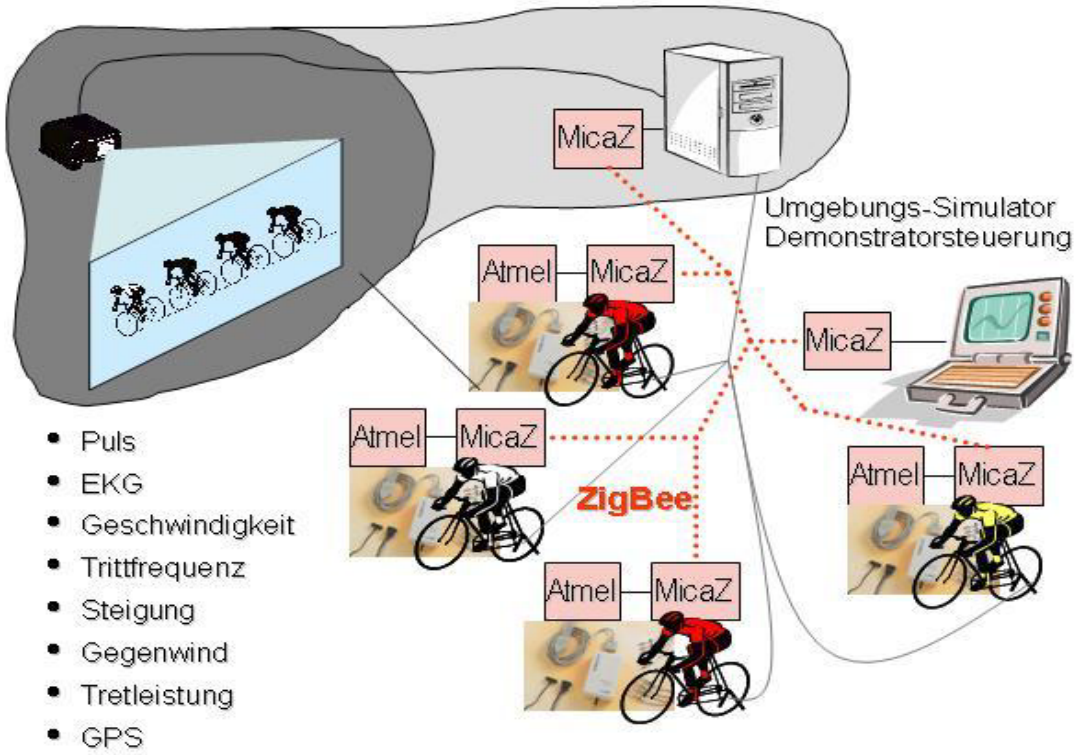


Abbildung 2.1: Fahrrad-Demonstrator

Weiterhin ist es möglich, dass der Fahrrad-Demonstrator sowohl im Freien ('Real-life Demonstrator') als auch im Raum ('Indoor-Demonstrator') eingesetzt werden kann. Für Letzteres sind die Fahrräder mit zusätzlichen Ergotrainer-Rollenständen, einer entsprechenden Umgebungssimulation und Visualisierung ausgestattet. Der Simulator wird hier unter anderem die Bremskräfte über den Ergotrainer vorgeben. Diese Masterarbeit bezieht sich nur auf ein Fahrrad des Indoor-Demonstrators und darauf, eine Herzfrequenz-Regelung des Radfahrers zu entwerfen. Die im Folgenden aufgelisteten Forderungen müssen von dem zu entwerfenden System erfüllt werden:

- Die Sensordaten: Herzfrequenz, Trittfrequenz und Geschwindigkeit sollen gemessen werden, um diese in der Regelung zu verarbeiten.

- Die Sensordatenerfassung und Auswertung werden auf der im Rahmen von AmI entwickelten Sensorplatine, deren zentrale Einheit der Mikroprozessor ATmega128L der Firma Atmel ist, implementiert. Der Mikroprozessor wird von einem 7,3728-MHz-Quarz betrieben.

- Die Herzfrequenz- und Leistungsregler sollen auf einem PC in MATLAB/SIMULINK mit einer kaskadierten Regelungsstruktur realisiert werden (Herzfrequenzregelung mit unterlagerter Leistungsregelung).

- Der Datenaustausch zwischen PC und Sensorplatine wird über eine serielle UART (Universal Asynchronous Receiver/Transmitter) Schnittstelle realisiert.

- Ein PI-Regler ist für den Herzfrequenzregler zu verwenden. Der Regler soll die allgemeinen Spezifikationen eines Regelungssystems erfüllen. Außerdem soll er möglichst schnell und gut gedämpft sein.

- Eine Benutzerschnittstelle soll erstellt werden, um die aktuellen Trainingszustandsmeldungen darzustellen.

- Die entwickelte Realisierung ist so auszulegen, dass sie als Grundlage für weiterführende Forschungen dienen kann.

Auf der Basis dieser Forderungen ist zunächst ein Realisierungskonzept zu erarbeiten, welches versucht, die Forderungen möglichst optimal zu erfüllen. Dieses Konzept stellt eine verbale Beschreibung des später mit vorgegebenen Hardwaremodulen aufzubauenden Systems dar. Im Anschluss daran, muss die gefundene Beschreibung des zusammenhängenden Systems so verfeinert werden, dass sie in Hardware realisiert werden kann. Dann ist die Software zu erstellen und mit der vorgegebenen Hardware in die Lage zu versetzen, die geforderte Regelungsaufgabe zu erfüllen.

2.2 Systembeschreibung

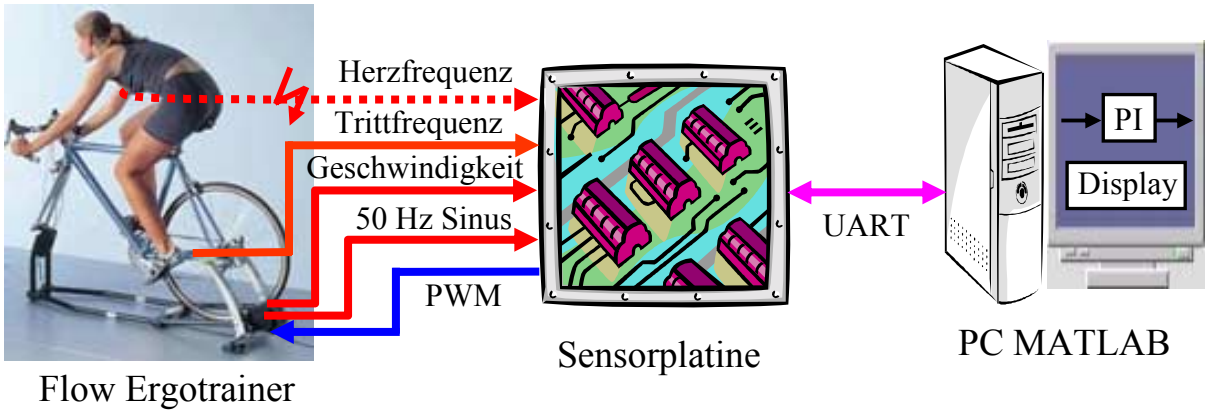


Abbildung 2.2: Aufbau des Indoor-Demonstrator-Systems

In Abbildung 2.2 ist der Aufbau des Indoor-Demonstrator-Systems zu sehen. Das System besteht aus einem PC mit MATLAB/SIMULINK, der Sensorplatine und dem Flow Ergotrainer der Firma Tacx, in den das Fahrrad eingespannt wird. Der Ergotrainer besteht aus einem zusammenklappbaren CycleForce-Rahmen und einer elektromagnetischen Wirbelstrombremse, in die der Geschwindigkeitssensor integriert ist.

Als Stellgröße des Leistungsregelkreises steht ein PWM-Signal (Pulse Weiten Moduliert) zur Verfügung. Dieses PWM-Signal ist proportional zum erzeugenden Gegenmoment der Wirbelstrombremse. Um die Wirbelstrombremse zu steuern, muss das PWM-Signal mit dem aus der Wirbelstrombremse kommenden Sinus-Signal synchronisiert werden. Die Synchronisation muss so erfolgen, dass eine positive Flanke des PWM-Signals nur beim Null-Durchgang des Sinus-Signals ausgegeben werden kann.

Es werden die Herzfrequenz des Radfahrers über Funk, dessen Trittfrequenz sowie die Geschwindigkeit des Rades über Kabel gemessen. Die Trittfrequenz des Radfahrers wird mit einem Trittmagnetsensor der Firma Tacx gemessen. Die Herzfrequenz des Radfahrers wird mit einem Pulssensor der AG Medion und einem im Rahmen von AmI entwickelten Empfänger über Funk gemessen. Die Geschwindigkeit wird mit dem in die Wirbelstrombremse integrierten Geschwindigkeitssensor gemessen. Die Erfassung und die Auswertung dieser Sensorsignale werden auf der Sensorplatine implementiert und daraufhin alle Sensorsignale über die serielle UART Schnittstelle an einen PC übergeben, auf welchem der Regelalgorithmus realisiert wird. Die bei dieser Masterarbeit verwendeten Hardwarekomponenten werden in den folgenden Kapiteln näher erläutert.

3 Verwendete Hardwarekomponenten

In diesem Kapitel werden alle in dieser Masterarbeit verwendeten Hardwarekomponenten und ihre Aufgaben vorgestellt. Zu Beginn wird der Flow Ergotrainer der Firma Tacx, der für den Indoor-Demonstrator zur Verfügung steht, kurz vorgestellt. Daran anschließend wird der Herzfrequenzsensor, die Pulsuhr LT-3680 der AG Medion, betrachtet. Zum Schluss werden die im Rahmen von AmI entwickelte Sensorplatine, auf der die Datenerfassung und Auswertung implementiert werden, sowie deren zentrale Einheit, der Mikroprozessor ATmega128L der Firma Atmel, vorgestellt.

3.1 Der Flow Ergotrainer der Firma Tacx

Abbildung 3.1: Der Flow Ergotrainer der Firma Tacx

Der Ergotrainer (siehe Abbildung 3.1) besteht aus dem Flow Computer und einem zusammenklappbaren CycleForce-Rahmen, in den das Fahrrad eingespannt werden kann. Der CycleForce Trainer ist standardmäßig für Renn- und Trekking-Räder, sowie für Mountainbikes mit einem Laufraddurchmesser zwischen 601 und 720 mm geeignet. Er ist mit einer elektromagnetischen Wirbelstrombremse, die auf dem CycleForce-Rahmen befestigt ist, ausgestattet. Die Wirbelstrombremse wird mit Netzspannung 230V/50Hz betrieben. Der große Vorteil dieses Bremssystems liegt darin, dass es sehr genau arbeitet und die Leistung exakt aufnimmt und wiedergibt. Die Bremsrolle wird durch einen einmalig einzustellenden Schnell-

spannhebel gegen den Fahrradreifen gepresst, wodurch der Druck der Bremseinheit auf den Reifen stets gleich ist und das Fahrrad schnell und einfach in den Trainer eingespannt werden kann. Die Bremse hat ein schweres, stählernes Schwungrad, welches ein gutes Trägheitsmoment garantiert. Der Geschwindigkeitssensor ist in die Bremse integriert, so dass der Raddurchmesser des Fahrrades nicht gesondert programmiert werden muss, weil die Geschwindigkeit der Bremsrolle gleich der Geschwindigkeit des Rades ist. Der Trittfrequenzsensor besteht aus einem Trittfrequenzmagnet und einem Magnetsensor. Für die Trittfrequenzmessung, sollen der Trittfrequenzmagnet an der Innenseite der linken Tretkurbel des Fahrrades und der Magnetsensor an der Unterseite des linken Hinterbaus befestigt werden, und zwar so, dass sich der Sensor in Höhe des Magneten befindet und dabei von diesem ungefähr 3 mm entfernt ist. Die gemessenen Signale werden über ein 6-adriges Kabel zur Sensorplatine gesendet, um sie dort auszuwerten und an einen PC weiterzuleiten. Für den Indoor-Demonstrator des AmI-Szenarios wird ein PC, auf dem die in dieser Masterarbeit entwickelten Regler implementiert und die gemessene Sensordaten visualisiert werden, den Flow Computer ersetzen. Die genauen technischen Daten des Ergotrainers können der Dokumentation des Herstellers [Web-03] entnommen werden.

3.2 Der Herzfrequenzsensor

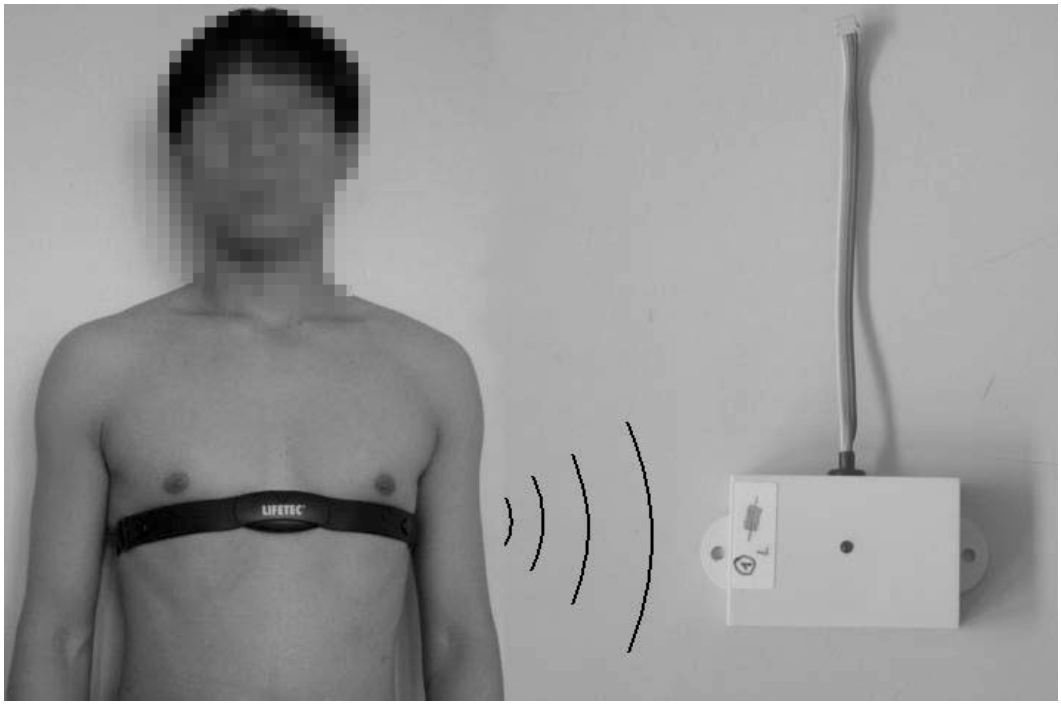


Abbildung 3.2: Pulsuhr LT 3680

Der Herzfrequenzsensor besteht aus einem Sender der AG Medion und einem im Rahmen von AmI entwickelten Empfänger. Der Sender wird auf der Haut getragen und sendet das Herzfrequenzsignal drahtlos zum Empfänger mit einer Sendefrequenz von 5,3 KHz. Das elastische Band des Senders sollte so eingestellt werden, dass der Sender eng an der Brust anliegt (siehe Abbildung 3.2 links). Um einen besseren Kontakt mit der Haut zu erreichen, sollte das

leitende Senderkissen z.B. mit EKG-Gel oder Wasser leicht befeuchtet werden. Bevor die Messung gestartet wird, sollte der Gurt an der Brust einige Minuten anliegen, um ihn auf die Körpertemperatur zu erwärmen. Um ein besseres Messergebnis zu erzielen, muss eine gute Position für den Gurt gefunden werden, so dass der Kontakt zwischen Haut und Sender bei tiefen Atemzügen nicht unterbrochen wird. Um korrekte Pulssignale zu erreichen, sollte einen Abstand von ca. 2 Meter zu anderen Funkgeräten eingehalten werden. Der Empfänger empfängt das Herzfrequenzsignal drahtlos innerhalb eines Abstandes von 1 Meter zum Sender. Wenn er das Pulssignal fehlerfrei empfängt, blinkt die LED (Leucht-Diode) in dem Empfänger-Gehäuse mit der selben Frequenz (siehe Abbildung 3.2 rechts) und leitet das Pulssignal zur Sensorplatine. Die genauen technischen Daten des Herzfrequenzsensors können der Dokumentation des Herstellers [Web-02] entnommen werden.

3.3 Die Sensorplatine und der ATmega128L

Abbildung 3.3: Sensorplatine

Die Sensorplatine (siehe Abbildung 3.3) dient als Entwicklungs-Board für die Erfassung und Auswertung der Sensordaten im Fahrrad-Demonstrator. Über die UART Schnittstelle können die erfassten Sensordaten an einen PC, der als Simulator arbeitet, oder an einen MicaZ-Knoten gesendet werden, der eine Weiterleitung über ZigBee ermöglicht.

Die Ausgangsdaten der Sensoren werden in externen Zählern verarbeitet. Mit den 8-Bit-Zählerbausteinen 74HC590 sind, je nach Bedarf, 8-, 16- oder 24-Bit-Zähler für die Sensoren aufgebaut. Die Ausgänge der 8-Bit-Zählerbausteine sind über den Daten-Bus DB(0:7) mit dem ATmega128L verbunden. An diesen Bus ist noch ein zweizeiliges ASCII (American Standard Code for Information Interchange)-LCD (Liquid Crystal Display)-Display, eine ein-

fache 8-Bit-Ausgabe und eine einfache 8-Bit-Eingabe angeschlossen. Weiter kann noch über eine zweite UART-Schnittstelle ein GPS-Modul (Global Positioning System) angeschlossen werden.

Die zentrale Einheit der Sensorplatine ist ein Mikroprozessor der Firma Atmel. Dieser hat die Aufgabe alle Funktionen der Sensorplatine auszuführen. Es handelt sich hierbei um den Mikroprozessor ATmega128L, welcher zur 8-Bit AVR RISC-Serie (Reduced Instuction Set Computer) gehört. Diese Serie zeichnet sich trotz der RISC-Architektur durch einen mächtigen Befehlssatz von 133 Befehlen aus. Diese sind für die Verwendung von Hochsprachencompilern optimiert. Daher ist es ohne großen Overhead im Programmcode möglich, die Prozessoren dieser Serie in C oder einer anderen Hochsprache zu programmieren. In dieser Anwendung wird der Prozessor von einem Quarz mit 7,3728 MHz betrieben. Das heißt, es können 7372800 Befehle pro Sekunde abgearbeitet werden. Weiterhin verfügt der ATmega128L über 53 Ein-/Ausgabekanäle, welche als binäre Verbindungen zur Umgebung dienen. Ebenso sind sehr viele Sonderfunktionen anstatt der binären E/A-Funktion aktivierbar. Eine wichtige Sonderfunktion, die im Rahmen der Sensorplatine eine sehr wichtige Aufgabe erfüllt, ist die USART (Universal Synchronous/Asynchronous Receiver/Transmitter) Schnittstelle. Die stellt eine serielle synchrone/asynchrone Verbindung zur Verfügung, die hier zum Datenaustausch zwischen Mikroprozessor und PC verwendet wird. Die genaue Funktion und Programmierung wird später erläutert. Im Folgenden sind einige der wichtigsten peripheren und internen Funktionen des Prozessors aufgelistet:

- zwei programmierbare USART-Schnittstellen
- ein 8-Kanal 10-Bit Analog-Digital-Umsetzer
- zwei 8-Bit Timer/Counter
- zwei 16-Bit Timer/Counter
- zwei 8-Bit pulsweitenmodulierte Ausgabekanäle
- bis zu sechs pulsweitenmodulierte Ausgabekanäle mit einstellbarer Auflösung(2···16-Bit)
- 128K Bytes reprogrammierbar In-System-Flash
- 4K Bytes Interne SRAM (Static Random Access Memory)
- ein 4-KByte EEPROM (Electrically Erasable Programmable Read Only Memory) zur Datensicherung
- eine byteorientierte SPI (Serial Peripheral Interface) Schnittstelle für das In-System-Programmierung
- programmierbarer Watchdog-Timer

Eine genaue Beschreibung des Mikrocontrollers kann dem Datenblatt [Atm-03] des Herstellers entnommen werden.

4 Konzeptentwurf des Regelungssystems

In diesem Kapitel wird aus den in der Aufgabenstellung genannten Vorgaben und den zur Verfügung stehenden Resourcen ein Konzept entwickelt, das alle Vorgaben erfüllt. Durch die Aufgabenstellung ist unter anderem festgelegt, dass es zwei zusammenhängende Regelkreise gibt. Zunächst wird das physikalische Ersatzbild des Indoor-Demonstrators vorgestellt.

4.1 Physikalisches Ersatzbild des Indoor-Demonstrators

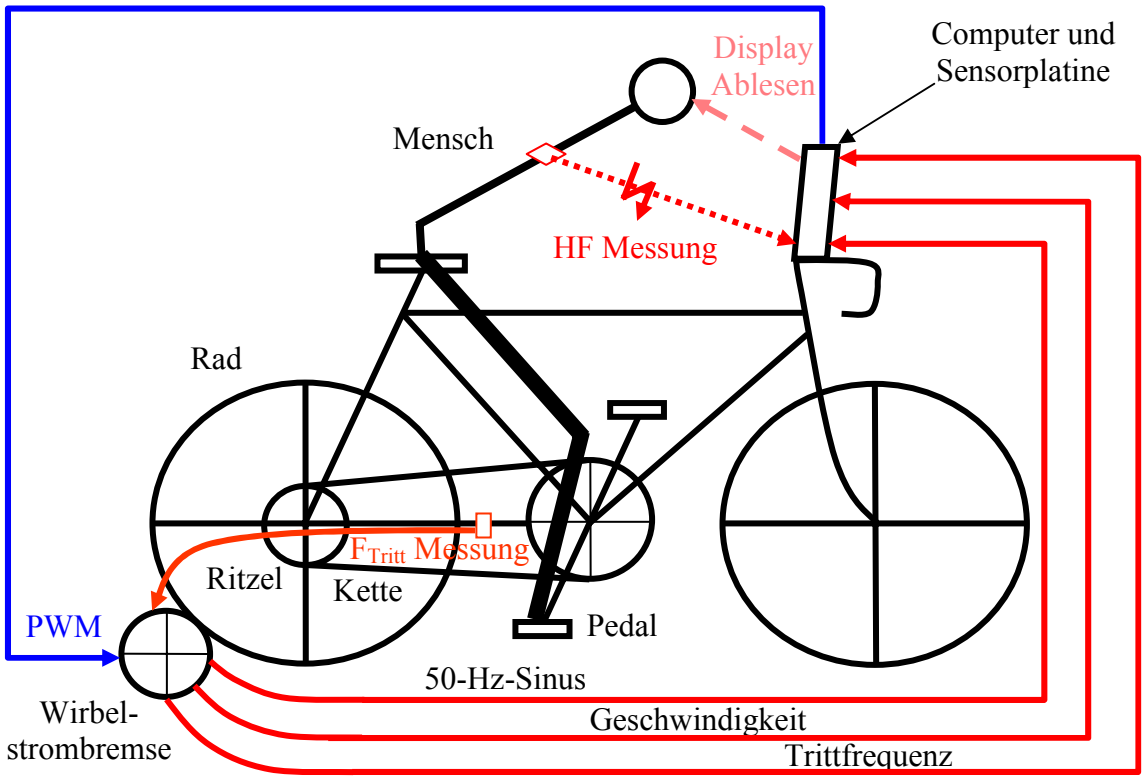


Abbildung 4.1: Physikalisches Ersatzbild des Indoor-Demonstrators

In Abbildung 4.1 ist das physikalische Ersatzbild des Indoor-Demonstrators dargestellt. Der Fahrer sitzt auf dem Ergotrainer und trainiert mit den vom Trainer vorgegebenen Trainingskriterien. Die Tritt- und Herzfrequenz des Fahrers sowie die Geschwindigkeit des Rades werden gemessen und an einen Computer, auf dem die Regler implementiert sind, gesendet. Der Leistungsregler generiert anhand der Sensordaten ein PWM-Signal, um die Wirbelstrombremse so zu steuern, dass die Abweichung der aktuellen Herzfrequenz von der Zielfrequenz

minimiert wird. Um die Wirbelstrombremse zu steuern, wird das PWM-Signal mit dem aus der Bremse kommenden 50-Hz-Sinus-Signal synchronisiert.

Alle Sensordaten sowie die Soll-Größen werden als Trainingszustandsmeldungen auf dem Display dargestellt. Während des Trainings liest der Fahrer die Trainingszustandmeldungen ab und hat die Möglichkeit, die Übersetzung des Fahrrades so umzuschalten, dass er die geforderte Leistung erbringt und die Wirbelstrombremse im zulässigen Bereich arbeitet.

4.2 Blockschaltbild des Regelungssystems

Das ganze Regelungssystem besteht aus zwei unterlagerten Regelkreisen. Außen ist der Herzfrequenzregelkreis, der den inneren Leistungsregelkreis überlagert. Das heißt, die Leistungsregelung ist die Basis der Herzfrequenzregelung. Dieses ist auch aus dem Zusammenhang aller Größen zu erkennen. Die Stellgröße des Herzfrequenzregelkreises dient gleichzeitig als Soll-Größe des Leistungsregelkreises. Beide Regelkreise haben den gleichen Ist-Wert, die aktuelle Herzfrequenz des Fahrers. Zunächst wird der Herzfrequenzregelkreis betrachtet.

4.2.1 Blockschaltbild des Herzfrequenzregelkreises

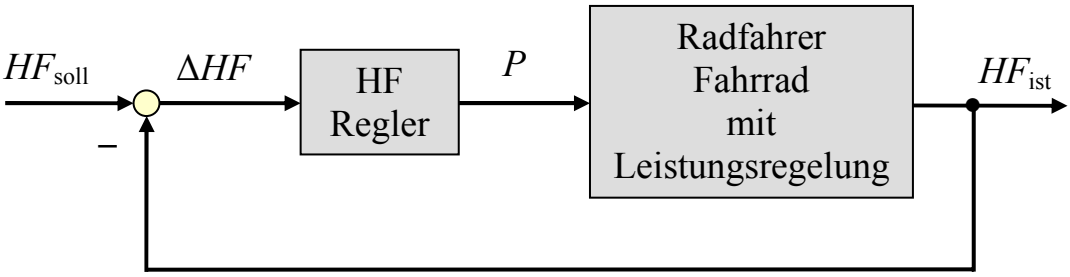


Abbildung 4.2: Blockschaltbild des Herzfrequenzregelkreises

In Abbildung 4.2 ist das Blockschaltbild des Herzfrequenzregelkreises dargestellt. Der Fahrer mit Fahrrad und Leistungsregelkreis wird hier als Strecke des Herzfrequenzregelkreises betrachtet. Beim Training gibt der Trainer eine Soll-Herzfrequenz HF_{soll} vor. Durch Vergleich mit der aktuellen Herzfrequenz HF_{ist}, die beim Fahrer gemessen wird, ergibt sich die Regeldifferenz ΔHF. Der Herzfrequenzregler reagiert auf die Regeldifferenz ΔHF und erzeugt entsprechend einen Leistungswert P als Stellgröße des Herzfrequenzregelkreises, der gleichzeitig als Soll-Größe des Leistungsregelkreises dient. Wegen der Beschränkung der menschlichen Leistung, kann die Stellgröße P nicht beliebig groß gewählt werden. Natürlich ist die menschliche Leistung immer positiv. Deshalb wird die Leistung in dieser Masterarbeit im Bereich von 10 Watt bis 1150 Watt beschränkt. Für den Herzfrequenzregler, wird ein klassischer PI-Regler verwendet.

4.2.2 Blockschaltbild des Leistungsregelkreises

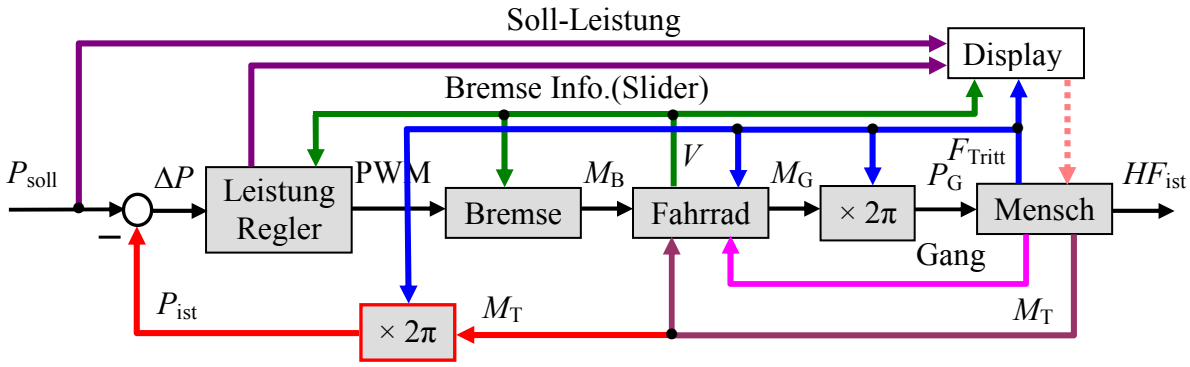


Abbildung 4.3: Blockschaltbild des Leistungsregelkreises

In Abbildung 4.3 ist das Blockschaltbild des Leistungsregelkreises dargestellt. In diesem Regelkreis ist der Mensch mit dem Fahrrad die Strecke und die Wirbelstrombremse der Aktuator. Die vom Herzfrequenzregler erzeugte Leistung P dient hier als Soll-Größe P_{soll}. Durch Vergleich mit der Ist-Leistung P_{ist}, ergibt sich die Regeldifferenz ΔP. Der Leistungsregler reagiert auf die Regeldifferenz ΔP und erzeugt entsprechend ein PWM-Signal als Stellgröße, das proportional zum Gegenmoment der Bremse ist. Die Bremse erzeugt anhand der Pulsbreite des PWM-Signals und der Geschwindigkeit ein Moment M_B an dem Fahrrad. Durch die Übersetzung des Fahrrades, ergibt sich von M_B ein Gegenmoment M_G am Pedal. M_B mal 2π mal Trittfrequenz F_{Tritt} ergibt die Gegenleistung P_G. Der Mensch tritt mit der Frequenz F_{Tritt} in die Pedale und erzeugt das Trittmoment M_T, um die Gegenleistung P_G zu kompensieren. Alle Soll-Größen und Ist-Größen werden als Trainingszustandsmeldungen auf dem Display dargestellt. Während des Trainings, liest der Mensch die Trainingszustandsmeldungen ab und hat die Möglichkeit, die Übersetzung des Fahrrades so umzuschalten, dass er die geforderte Leistung erbringt und die Wirbelstrombremse im zulässigen Bereich arbeitet. Aufgrund der fehlenden Möglichkeit die Ist-Leistung P_{ist} oder das Trittmoment M_T direkt zu messen, ist die Leistungsregelung mit dieser Regelungsstruktur zurzeit nicht realisierbar. Deshalb muss der Leistungsregelkreis korrigiert werden.

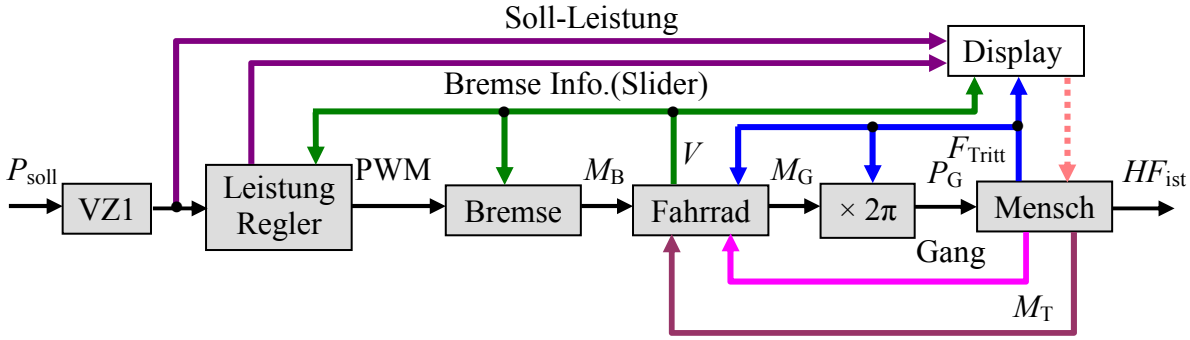


Abbildung 4.4: Blockschaltbild des realisierbaren Leistungsregelkreises

Der korrigierte realisierbare Leistungsregelkreis ist in Abbildung 4.4 zu sehen. Es ist in diesem Regelkreis keine direkte Rückkopplung der Ist-Leistung vorhanden. Dafür wird ein VZ1-Glied (Verzögerungsglied 1. Ordnung) vor den Leistungsregler hinzugefügt, um die starke

Änderung der Soll-Leistung zu glätten. Das VZ1 Glied hat noch eine weitere wichtige Funktion im gesamten Regelkreis, da es die algebraische Schleife des gesamten Regelkreises unterbricht.

Die Pulsbreite des PWM-Signals hängt von Leistung und Drehzahl der Wirbelstrombremse ab. Die Drehzahl mal Umfang der Bremsrolle ist die Geschwindigkeit der Bremsrolle. Das heißt, die Pulsbreite des PWM-Signals ist von Leistung und Geschwindigkeit der Bremse abhängig. Das bedeutet, dass in diesem korrigierten Leistungsregelkreis der Leistungsregler zwei Eingänge hat: die Geschwindigkeit V und die Leistung P_{soll}. Diese Aufgabe ist mit einem normalen SISO-Regler (Single Input Single Output) nicht realisierbar. Deshalb wird in dieser Arbeit der Leistungsregler durch einen zwei-dimensionalen Lookup-Table realisiert. Dadurch ist die Leistungsregelung eigentlich eine Leistungssteuerung. Dieser zwei-dimensionale Lookup-Table beschreibt die Eigenschaften der Wirbelstrombremse und wird später im Abschnitt 6.2.2.1 *Identifikation der Wirbelstrombremse* genauer vorgestellt.

4.3 Sensordatenerfassung und Auswertung

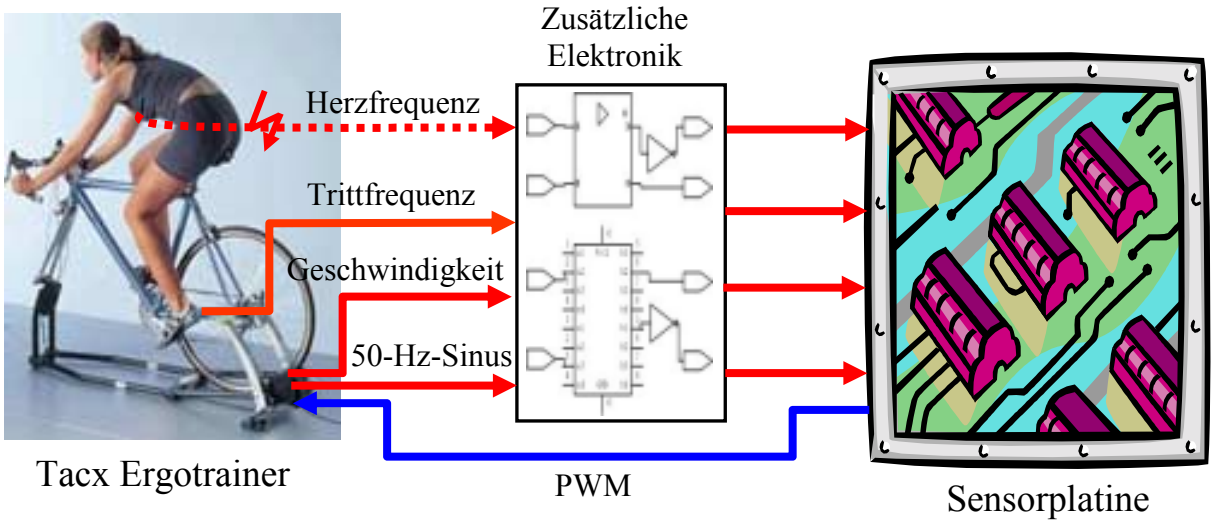


Abbildung 4.5: Übersicht der Sensordatenerfassung

Wie in der Aufgabenstellung schon erwähnt, werden die Herzfrequenz und Trittfrequenz des Fahrers sowie die Geschwindigkeit des Rades gemessen. Die Sensordatenerfassung und Auswertung werden auf der im Rahmen von AmI entwickelten Sensorplatine implementiert. Doch bevor die Messung starten kann, müssen die Sensorsignale in einer zusätzlichen Elektronikplatine verarbeitet werden. Danach können sie an die Zählerbausteine der Sensorplatine angeschlossen werden. Die zusätzliche Elektronikplatine (siehe Abbildung 4.5) hat folgende Funktionen:

1. Das Rauschen der Sensorsignale soll herausgefiltert werden

2. Die Flanken der Sensorsignale sollen steiler geformt werden

3. Die Herz- und Trittfrequenzsignale sowie das 50-Hz-Sinus-Signal sollen in Rechtecksignale umgewandelt werden

Die Umwandlung der Herz- und Trittfrequenzsignale werden im nächsten Abschnitt ausführlich erklärt. Hier wird die Umwandlung des Sinus-Signals kurz erläutert. Um die Wirbelstrombremse zu steuern, kann eine positive Flanke des PWM-Signals nur beim Null-Durchgang des Sinus-Signals ausgegeben werden. Deshalb wird das Sinus-Signal in Rechtecksignal so umgewandelt, dass bei jedem Null-Durchgang des Sinus-Signals ein Flankenwechsel des Rechtecksignals stattfindet (siehe Abbildung 4.6).

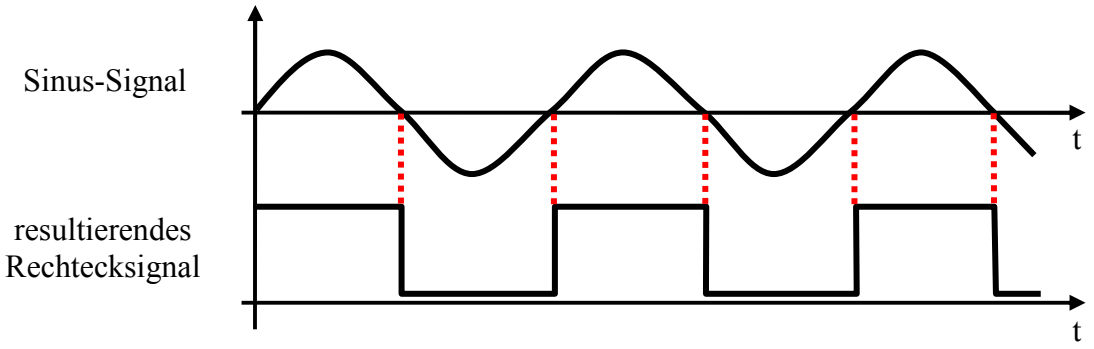


Abbildung 4.6: Umwandlung des Sinus-Signals

Nach der Umwandlung des Sinus-Signals in ein Rechtecksignal, können die positiven Flanken des PWM-Signals anhand der Flanken des resultierenden Rechtecksignals ausgegeben werden.

Für die richtige Messung aller Sensordaten, sind folgende Vorgaben zu erfüllen:

- Der Mikroprozessor ATmega128L, die zentrale Einheit der Sensorplatine, wird von einem 7,3728-MHz-Quarz betrieben.
- Je nach Bedarf, steht ein 8-, 16- oder 24-Bit-Zähler für die Sensoren zur Verfügung.
- Die Trittfrequenz eines Radsportlers liegt zwischen 0 und 220 U/min (Umdrehungen pro Minute).
- Die Herzfrequenz eines Radsportlers liegt zwischen 40 und 220 S/min (Schläge pro Minute).
- Die Geschwindigkeit von einem Fahrrad im Radsport auf die flache Strecke beträgt von 0 bis 80 km/h (Kilometer pro Stunde).

Im Folgenden wird unter diesen Vorgaben versucht eine Konzeption der Realisierung der Datenerfassung zu entwerfen. Als Erstes wird hierzu die Herz- und Trittfrequenzmessung betrachtet.

4.3.1 Herz- und Trittfrequenzmessung

Aus den obigen Vorgaben ist ersichtlich, dass die Herz- und Trittfrequenz des Radfahrers dem gleichen Wertebereich haben. Damit können beide Signale mit einem gleichen Verfahren ge-

messen werden. Die Signale liegen im Bereich von 0 bis 220 U/min bzw. S/min, also im Frequenzbereich kleiner 4 Hz. Um diese Sensorsignale genau zu messen, wird die Zeitdauer zwischen zwei Impulsen der Sensorsignale im Zähler gemessen, und anschließend in die Frequenz umgerechnet. Dafür werden zuerst die Sensorsignale in Rechtecksignale umgewandelt.

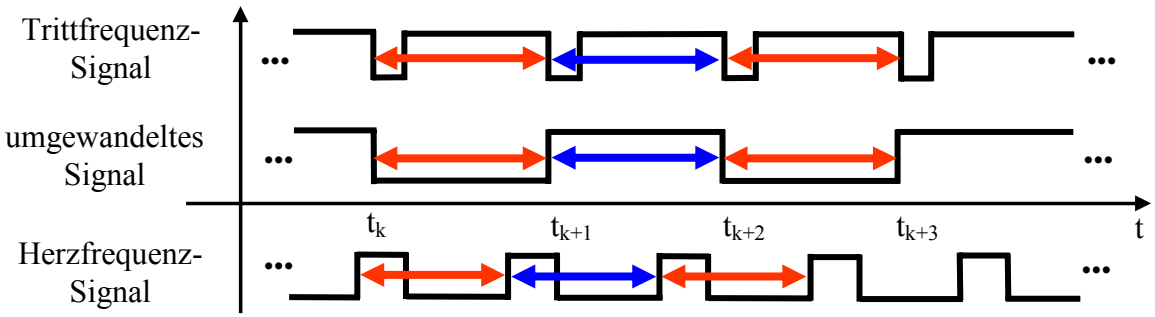


Abbildung 4.7: Tritt- und Herzfrequenz: Signalform und Umwandlung

Die Trittfrequenzsignale sind negative Impulse und die Herzfrequenzsignale positive Impulse wie in Abbildung 4.7 dargestellt. Die Umwandlung erfolgt so, dass die Flanke des Rechtecksignals nach jeder negativen Flanke des Trittfrequenzsignals bzw. positiven Flanke des Herzfrequenzsignals einmal umgekehrt wird. Danach ist die Zeitdauer zwischen zwei Impulsen der Sensorsignale gleich der Pulsdauer der Rechtecksignale: $\Delta t = t_{k+1} - t_k$.

Für die Messungen beider Sensorsignale werden jeweils drei 8-Bit-Zähler kaskadiert verwendet. Das Rechtecksignal wird am Clock-Enable-Pin des ersten Zählers angeschlossen. Wenn eine negative Flanke des Trittfrequenzsignals bzw. positive Flanke des Herzfrequenzsignals auftritt, ändert sich der Pegel des Rechtecksignals auf "niedrig" und die Zähler beginnen zu zählen. Die Zähler laufen solange bis die nächste negative Flanke des Trittfrequenzsignals bzw. positive Flanke des Herzfrequenzsignals anliegt. Dann ändert sich der Pegel des Rechtecksignals auf "hoch" und die Zähler stoppen. Daraufhin kann der Zählerzustand ausgelesen werden. Der aktuelle Zählerzustand $\Delta_{Zähler}$ wird durch Subtraktion des alten Zählerzustandes von dem neuen Zählerzustand berechnet (Formel (4.1)).

$$\Delta_{Zähler} = Z(t_{k+1}) - Z(t_k) \tag{4.1}$$

Mit einem Eingangstakt der Zählerbausteine von 7,3728 MHz, kann die Trittfrequenz F_{Tritt} in U/min bzw. Herzfrequenz in S/min nach Formel (4.2) berechnet.

$$F_{Tritt} = \frac{7{,}3728 \cdot 10^6}{\Delta_{Zähler}} \cdot 60 \text{ (U/min)} \tag{4.2}$$

Wenn der aktuelle Zählerzustand $\Delta_{Zähler}$ null ist, wird die Trittfrequenz bzw. Herzfrequenz nicht nach Formel (4.2) berechnet, sondern direkt eine Null zugewiesen. Dies wird später im Abschnitt 6.2.1.3 *Auslesen des Zählers der Sensordaten* noch genauer erklärt.

Mit drei kaskadierten 8-Bit-Zähler nach Gleichung (4.3) ergibt sich eine minimale Messgröße von ungefähr 27 U/min bzw. S/min.

$$\frac{7{,}3728 \cdot 10^6}{2^{3\cdot 8}} \cdot 60 \approx 27 \text{ U/min} \tag{4.3}$$

Mit dem oben vorgestellten Verfahren auf Basis der vorhandenen Sensorplatine kann leider nur jeder zweite Puls gemessen werden. Um dieses Problem zu lösen, werden folgende Vorschläge für die mögliche Hardwareverbesserung gegeben:

1. Durch Verwendung der doppelten Anzahl der Zählerbausteine, z.B. drei 8-Bit-Zähler zählen den "niedrigen" Pegel des Rechtecksignals und weitere drei 8-Bit-Zähler zählen den "hohen" Pegel.

2. Durch Verwendung der externen Hardwareunterbrechungen des Mikroprozessors können die Tritt- bzw. Herzfrequenzsignale ohne Umwandlungen direkt gemessen werden. Die Sensorsignale können direkt an den Pins der externen Hardwareunterbrechungen des Mikroprozessors angeschlossen werden. Wenn die Hardwareunterbrechung durch die Flanke des Sensorsignals ausgelöst wird, kann ein Zähler in der ISR (Interrupt Service Routine) der Hardwareunterbrechung gestartet werden, um die Zeitdauer dieses Impulses des Sensorsignals zu messen. Der Zähler zählt solange bis die nächste gleichartige Flanke eine weitere Unterbrechung auslöst. Danach kann aus dem Zählerzustand die aktuelle Tritt- bzw. Herzfrequenz berechnet werden.

4.3.2 Geschwindigkeitsmessung

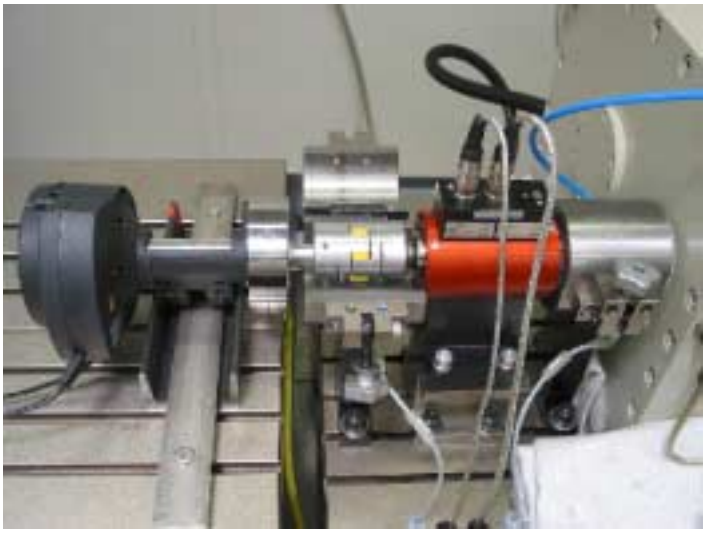


Abbildung 4.8: Messaufbau der Wirbelstrombremse

Als Nächstes wird das Konzept der Geschwindigkeitsmessung vorgestellt. Wie schon erläutert, ist die Geschwindigkeit des Rades gleich der Geschwindigkeit der Bremsrolle. Deshalb kann die Geschwindigkeit des Rades durch die Messung der Bremsrollengeschwindigkeit erfasst werden. Das Signal des in die Bremse integrierten Geschwindigkeitssensors ist ein Rechtecksignal mit wechselnder Frequenz. Um dieses Geschwindigkeitssignal zu messen, muss das Verhältnis zwischen der Signalfrequenz in Hz und der Geschwindigkeit in km/h bestimmt werden. Dafür wird die Wirbelstrombremse mit dem in Abbildung 4.8 dargestellten

Messaufbau mit einer Messwelle gekoppelt. Die Rolle der Wirbelstrombremse hat einen Durchmesser von D = 28,9 mm. Die Messung wurde mit unterschiedlichen Drehzahlen durchgeführt. Die Ergebnisse dieser Messung und die berechnete Faktoren sind in Tabelle 4.1 aufgeführt.

Drehzahl N (U/min)	500	1000	1500	2000
Signalfrequenz F (Hz)	33,3	66,7	100	133,3
Geschwindigkeit V (km/h)	2,7238	5,4475	8,1713	10,895
Faktor K ($\frac{Hz}{km/h}$)	12,226	12,244	12,238	12,235

Tabelle 4.1: Messergebnisse der Wirbelstrombremse

Die erste Zeile der Tabelle 4.1 beinhaltet die Drehzahlen von 500, 1000, 1500 und 2000 U/min. In der zweiten Zeile sind die gemessenen Signalfrequenzen in Hz notiert. Die in dritte Zeile eingetragenen Geschwindigkeitswerte in km/h werden aus den unterschiedlichen Drehzahlen nach Formel (4.4) bestimmt.

$$V(km/h) = \frac{N(U/\min)}{60} \cdot \pi \cdot D \tag{4.4}$$

In Formel (4.4) ist N die Drehzahl und D der Durchmesser der Bremsrolle. Die in der vierten Zeile eingetragenen Faktoren K werden aus der Signalfrequenz F, der Geschwindigkeit V und den unterschiedlichen Drehzahlen bestimmt. Es ist zu erkennen, dass sich für unterschiedliche Drehzahlen ein ähnlicher Faktorwert K ergibt. Für den in dieser Arbeit verwendeten Faktor K, wird der arithmetische Mittelwert der gemessenen Ergebnisse herangezogen. Aus Gleichung (4.5) beträgt der arithmetische Mittelwert $\overline{K}$ des Faktors einen Wert von $12{,}236\frac{Hz}{km/h}$.

$$\overline{K} = \frac{12{,}226 + 12{,}224 + 12{,}238 + 12{,}235}{4} \approx 12{,}236\,(\frac{Hz}{km/h}) \tag{4.5}$$

Wenn das Verhältnis zwischen der Signalfrequenz in Hz und der Geschwindigkeit in km/h bestimmt worden ist, kann die Geschwindigkeit in km/h nach Formel (4.6) berechnet werden.

$$V(km/h) = \frac{F(Hz)}{\overline{K}(\frac{Hz}{km/h})} = \frac{F(Hz)}{12{,}236(\frac{Hz}{km/h})} \tag{4.6}$$

Um die Geschwindigkeit V in km/h zu berechnen, muss die Frequenz F des Geschwindigkeitssignals gemessen werden. Aus den Vorgaben ist ersichtlich, dass die Geschwindigkeit eines Fahrrades im Radsport auf flacher Strecke einen Wert von 0 bis 80 km/h annimmt. Dies entspricht eine Signalfrequenz von 0 bis 978,88 Hz. Dafür kann die Messung der Signalfrequenz in Hz durch die Anzahlmessung der Signalimpulse in einer bestimmten Zeitdauer und anschließend durch Umrechnung in Hz erledigt werden (siehe Abbildung 4.9).

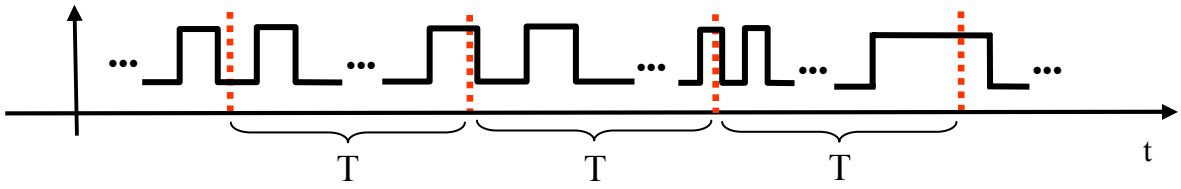


Abbildung 4.9: Messung der Frequenz des Geschwindigkeitssignals

Der aktuelle Zählerzustand $\Delta_{Zähler}$ wird durch Subtraktion des alten Zählerzustandes von dem neuen berechnet (Formel (4.7)).

$$\Delta_{Zähler} = Z(t+T) - Z(t) \tag{4.7}$$

Daraufhin kann die Signalfrequenz nach Formel (4.8) berechnet werden.

$$F(Hz) = \frac{\Delta_{Zähler}}{T} \tag{4.8}$$

Mit einem 8-Bit-Zähler, kann $\Delta_{Zähler}$ maximal einen Wert von $\Delta_{Zähler\_max} = 2^8 - 1 = 255$ betragen. Aus Gleichung (4.9) ist zu sehen, um die maximale Signalfrequenz von 978,88 Hz mit einem 8-Bit-Zähler richtig zu messen, benötigt es die maximale Zeitdauer T_{max} einen Wert von 0,2605 Sekunden.

$$T_{\max}(s) = \frac{\Delta_{Zähler\_\max}}{F_{\max}(Hz)} = \frac{255}{978{,}88(Hz)} = 0{,}2605(s) \tag{4.9}$$

In dieser Arbeit wird ein Wert von 0,25 Sekunden für die Zeitdauer T verwendet. Setzt man (4.8) und T in (4.6) ein, so erhält man die Formel für die Berechung der Geschwindigkeit in km/h.

$$V(km/h) = \frac{\Delta_{Zähler}}{12{,}236 \times 0{,}25} \tag{4.10}$$

Mit diesem Verfahren, beträgt der maximale Messfehler von $\Delta_{Zähler}$ 2 Impulse. Setzt man diese 2 Impulse in Formel (4.10) ein, so erhält man den maximalen absoluten Messfehler von ungefähr 0,65 km/h. Bei einer durchschnittlichen Trainingsgeschwindigkeit von 35 km/h, beträgt der maximale relative Messfehler ungefähr 1,86 %.

Auf der im Rahmen von AmI entwickelten Sensorplatine gibt es momentan nur einen Reset-Pin für alle Zählerbausteine. Das heißt, die Zähler können nach Auslesen des Inhaltes nicht einzeln zurückgesetzt werden. Somit kann ein Zählerüberlauf im Fall einer Fehlermessung nicht erkannt werden. Um dieses Problem zu lösen, kann je ein Reset-Pin für die Zähler der Trittfrequenz, der Herzfrequenz und der Geschwindigkeit verwendet werden. Hiermit können die Zähler jeweils nach Auslesen des Inhaltes zurückgesetzt werden, um Zählerüberläufe zu vermeiden.

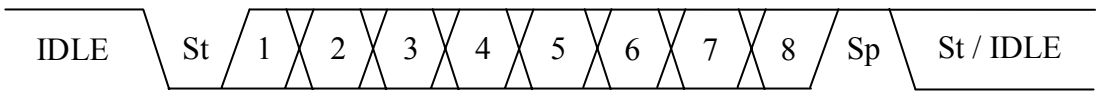
4.4 Die Schnittstelle

Die Schnittstellen des Indoor-Demonstrator-Systems ermöglichen den Informationsumtausch zwischen den Komponenten, die an dem System beteiligt sind. Es gibt drei Schnittstellen im System des Indoor-Demonstrators, die Signalschnittstelle zu den Sensoren, die PC- und die Benutzerschnittstellen. Die Signalschnittstelle zwischen den Sensoren und dem Mikroprozessor stellt die Ein- und Ausgänge der Sensorsignale dar. Sie ist durch den Entwurf der Sensorplatine festgelegt worden und wird hier nicht betrachtet. In dieser Arbeit, werden ausschließlich die PC- und die Benutzerschnittstellen betrachtet. Zuerst wird auf den Konzeptentwurf der PC-Schnittstelle eingegangen.

4.4.1 Die PC-Schnittstelle

Die Aufgabenstellung fordert, dass alle Sensordaten vom Mikroprozessor an einen PC und die Stellgröße vom PC an den Mikroprozessor zu übertragen sind. Ebenfalls soll die Schnittstelle einen so hohen Datendurchsatz ermöglichen, dass der PC in einer späteren Arbeit auch Simulatorfunktionen wie z.B. Bergabfahren und Windeffekt übernehmen kann. Um Periphere I/O Pins zu sparen, wurde die serielle USART Schnittstelle des ATmega128L beim Sensorplatinenentwurf schon entsprechend festgelegt. Hierzu wird in dieser Arbeit nur die serielle Schnittstelle untersucht. Weitere Schnittstellen wie zum Beispiel die Parallele Schnittstelle oder USB (Universal Serial Bus) werden nicht betrachtet, da die zusätzliche notwendige Hardware und Software den Rahmen dieser Arbeit sprengen würde.

Die ausgewählte Schnittstelle zeichnet sich dadurch aus, dass die Daten seriell übertragen werden und deshalb nur wenige Pins des Mikrocontrollers belegen. Auch hat der Mikroprozessor ATmega128L ein entsprechendes USART Hardwaremodul, das den seriellen Datenfluss steuern kann. Mit Ausnutzung des USART Hardwaremoduls wird die Prozessorbelastung durch eine solche Kommunikation nicht relevant erhöht. Der maximal erreichbare Datendurchsatz mit zulässigem Fehler hat bei einem Quarz von 7,3728 MHz eine Baudrate von 230,4 kbps (Kilo Bit pro Sekunde). Mit diesem Datendurchsatz wäre es kein Problem, alle Anforderungen der Datenübertragung zu erfüllen. Allerdings muss dieser ebenfalls in der Lage sein, den entsprechenden Datendurchsatz zu erbringen und die anfallenden Daten entsprechend schnell zu verarbeiten. Hierzu wird zuerst den in Abbildung 4.10 gezeichneten Kommunikations-Datenframe aufgebaut.



St : Start Bit, immer niedrig.

n : Datenbits (0 bis 8-Bit).

Sp : Stop Bit, immer hoch.

IDLE : Keine Datenübertragung, muss immer hoch sein.

Abbildung 4.10: Format des Datenframes

Das Kommunikations-Datenframe hat eine Länge von zehn Bits, die sich folgendermaßen verteilen: ein Start Bit, acht Datenbits und ein Stop Bit. Dadurch kann die Übertragungseffizienz wie in die Formel (4.11) berechnet werden.

$$E = \frac{8 Datenbits}{10 Framebits} \cdot 100\% = 80\% \tag{4.11}$$

In Abbildung 4.11 ist die serielle UART Schnittstelle zwischen PC und Sensorplatine dargestellt. Die Trittfrequenz, Herzfrequenz und Geschwindigkeit werden von der Sensorplatine an den PC übertragen. Sie werden auf dem PC zur Regelung und Darstellung auf dem Display verwendet. Der von dem Leistungsregler erzeugte Indexwert der PWM-Tabelle wird vom PC an die Sensorplatine gesendet, um dort durch Umwandlung in ein PWM-Signal als Stellgröße zur Steuerung der Wirbelstrombremse zu dienen.

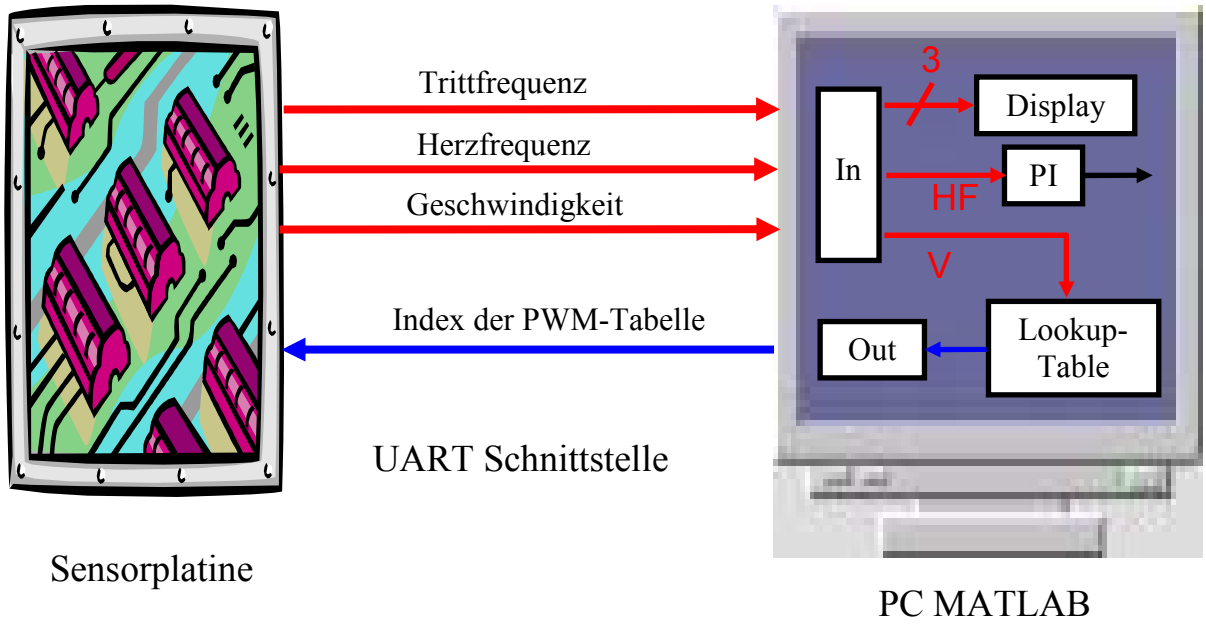


Abbildung 4.11: Datenaustausch zwischen PC und Sensorplatine

Auf der Basis des in Abbildung 4.10 dargestellten Datenframes kann das Übertragungsprotokoll folgendermaßen aufgebaut werden.

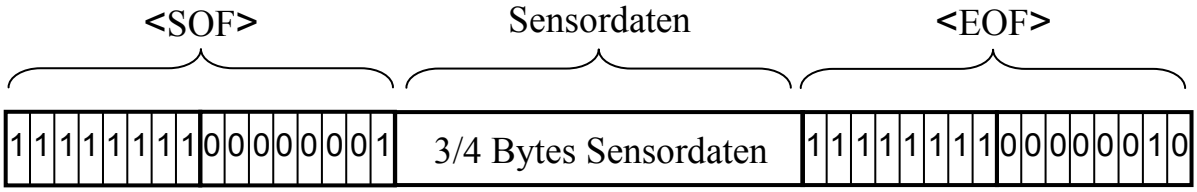


Abbildung 4.12: Protokollformat der UART-Schnittstelle

0xFF wird als Escape-Sequenz für <SOF> und <EOF> verwendet.
Jedem sonstigen Vorkommen von 0xFF wird ein 0xFF vorangestellt.

SOF : Start of Frame
2 Bytes: 0xFF 0x01

EOF : End of Frame
2 Bytes: 0xFF 0x02

Sensordaten : drei oder vier Bytes mit festen Reihenfolge
1.Byte: Herzfrequenz in S/min
2.Byte: Inhalt des Geschwindigkeitszählers, wenn der Wert 0xFF ist, werden zwei 0xFF hintereinander gesendet
Letztes Byte: Trittfrequenz in U/min

Durch das in der Abbildung 4.12 festgelegte Protokollformat ist es nun möglich, die notwendige minimale Datenrate für die Übertragung nach Gleichung (4.12) zu berechnen.

$$Datenrate_{\min} = \frac{N_P \cdot N_{Bit}}{T_A} \tag{4.12}$$

Dabei ist N_P gleich der Anzahl der zu sendenden Protokolle pro Abtastzeit, N_{Bit} gleich der Bits pro Protokoll und T_A gleich der Abtastzeit. Mit $N_P = 1$, maximal $N_{Bit} = 80$ bit und $T_A = 0{,}25$ s ergibt sich die ideale minimale Datenrate zu 320 bps. Allerdings ist zu beachten, dass das PWM-Signal als Stellgröße vor der Ausgabe zur Wirbelstrombremse vom PC empfangen werden muss. Das PWM-Signal wird in die ISR einer Hardwareunterbrechung mit einer Periode von 10 ms ausgegeben (siehe Abschnitt 6.2.1.4 *ISRs der Unterbrechungen*). Das heißt, die Dauer der Datenübertragung für einen Datenframe muss kleiner als 10 ms sein. Außerdem ist noch zu beachten, dass diese Dauer der Datenübertragung von 10 ms sich nur auf den reinen Empfangsbetrieb ohne Senden bezieht. Mit Berücksichtigung des Sendebetriebs muss die Datenübertragung für einen Datenframe kleiner als 5 ms sein. Es ist noch zu beachten, dass die Zeit für die Betriebsartenumschaltung während der Datenübertragung berücksichtigt werden muss. Weiterhin muss ein Sicherheitsabstand zwischen zwei Übertragungen eingehalten werden. Ebenso benötigen die Sensordatenerfassung und Berechnung der neuen Stellgrößen Zeit. Außerdem sollte die Zeit, die für die Ausführungen der ISRs aller Unterbrechungen des Mikroprozessors benötigt wird, auch berücksichtigt werden. So wird die Zeitdauer der Datenübertragung für einen Datenframe auf 3 ms abgeschätzt. Damit kann die tatsächlich notwendige Datenrate nach folgender Formel berechnet werden:

$$Datenrate_{real}(Bit/s) = \frac{N_{Frame}(bit)}{T_{Frame}(s)} \tag{4.13}$$

N_{Frame} ist die Angabe der Bits pro Datenframe und T_{Frame} die Übertragungszeit für einen Datenframe. Mit $N_{Frame} = 10$ bit und $T_{Frame} = 3$ ms ergibt sich die notwendige Datenrate zu ungefähr 33,33 kbps. Die nächst höhere auf dem Mikroprozessor mit zulässigem Fehler erreichbare Datenrate beträgt 38,4 kbps. Diese ist sowohl auf dem Mikroprozessor als auch auf dem PC erreichbar.

4.4.2 Die Benutzerschnittstelle

Eine weitere Forderung der Aufgabenstellung besagt, dass es möglich sein soll, Trainingszustandsmeldungen auf dem Display darzustellen. Die Sensorplatine verfügt zwar über drei LED und ein LCD, welche zu Signalisierung von Zuständen, aber nicht zur Ausgabe von Sensorsignalen oder Stellgrößen geeignet sind. Daher wird eine Benutzerschnittstelle zur Darstellung von Trainingszustandsmeldungen und Warnmeldungen vorgesehen. Es soll auch möglich sein, dass der Trainer oder Fahrer seine erwünschten Soll-Größen durch ein Eingabefenster eingeben kann. So wird eine von MATLAB unterstützte benutzerfreundliche GUI-Schnittstelle (Graphical User Interface) und ein von SIMULINK unterstütztes Eingabefenster erstellt. Auf die GUI-Schnittstelle werden die Warnmeldungen und Trainingszustandsmeldungen wie z.B. Tritt- und Herzfrequenz des Fahrers, Soll- und Ist-Leistung des Fahrers, Geschwindigkeit des Rads sowie Bremsstärke dargestellt. Das Eingabefenster dient zur Eingabe der Trainingssoll-Größen durch den Trainer während des Trainings, wie Soll-Leistung, Soll-Herzfrequenz und Soll-Trittfrequenz.

4.5 Zusammenfassung

In den letzten Abschnitten wurde aus den Vorgaben der Aufgabenstellung eine mögliche Realisierung des Herzfrequenz-Regelungssystems erarbeitet. Dabei wurde zuerst aus den Vorgaben das Blockschaltbild des gesamten Regelungssystems erstellt. Im Anschluss daran wurden die Datenerfassungsverfahren für verschiedene Sensordaten erklärt. Schließlich wurde ein bezüglich der Sensorplatine geeignetes Übertragungsprotokoll erarbeitet. Zum Schluss wurden die in dem Regelungssystem verwendeten Schnittstellen vorgestellt. Die Wahl der PC-Schnittstelle und Festlegung der Benutzerschnittstelle wurden frei getroffen. Dabei ist allerdings zu beachten, dass bei der kompletten Realisierung immer auch der Aspekt beachtet werden musste, dass das zu entwerfende System zu Forschungszwecken verwendet werden soll. Das heißt, dass alle Komponenten eher etwas überdimensioniert ausgelegt werden, um spätere Erweiterungen auf der gleichen Hardwareplattform zu ermöglichen.

5 Entwurf des Herzfrequenzreglers

In diesem Kapitel wird der in der Aufgabenstellung vorgestellte PI-Regler für den Indoor-Demonstrator nach dem in Abbildung 4.2 angezeigten Blockschaltbild entworfen. Dafür wird das computergestützte Entwurfsverfahren mit Hilfe von Wurzelortskurve (WOK) und Frequenzkennlinie (FKL) verwendet. Dieses Verfahren ist ein modellbasiertes Entwurfsverfahren, wobei zunächst ein lineares mathematisches Modell für das als Strecke betrachtete Mensch-Fahrrad-System identifiziert werden soll. Dieses Modell wird durch seine Übertragungsfunktion dargestellt und dient als Streckenmodell für den Regelungsentwurf. Als Erstes wird die Modellierung des Mensch-Fahrrad-Systems betrachtet.

5.1 Mensch-Modellierung

Das mathematische Modell ist die Beschreibung von Zusammenhängen zwischen dem System und seiner Umgebung. In dieser Arbeit soll zuerst ein mathematisches Mensch-Modell erstellt werden, das einen Zusammenhang zwischen der Leistung des Radfahrers und seiner Herzfrequenz beschreibt. Hier wird kurz auf die Modellbildung- und Identifikationsverfahren eingegangen. Auf die genaue Beschreibung der Modellbildung und Identifikation wird dabei verzichtet, da diese bei der Masterarbeit nicht von primärem Interesse ist und als bekannt vorausgesetzt werden darf. Alle Informationen über die mathematische Modellbildung können der Vorlesung “Modellbildung und Identifikation“ vom Professor L. Litz entnommen werden.

5.1.1 Grundlagen

Mathematische Modelle technischer Prozesse werden erstellt, um beim Regelungsentwurf modellbasierte Entwurfsverfahren anwenden zu können. In diesem Fall spricht man von einem *Synthesemodell*. Man unterscheidet prinzipiell zwei Wege, um zu einem mathematischen Prozessmodell zu gelangen:

1) Bei der *theoretischen Analyse* erhält man das mathematische Modell durch Anwendung physikalischer Gesetze. Typisch dabei ist das Aufstellen von Bilanzgleichungen für die Erhaltungsgrößen des Systems.

2) Bei der *experimentellen Analyse* werden Testsignale auf die Eingänge des Systems geschaltet und die daraus resultierenden Ausgangssignale gemessen. Aus den Testsignalen, den gemessenen Ausgangssignalen und einem vorher gewählten Modellansatz wird das mathematische Modell mit Hilfe eines Identifikationsverfahrens parametrisiert.

Um das mathematische Modell zu parametrisieren unterscheidet man prinzipiell zwei Identifikationsverfahren, dies sind das *deterministische* und das *stochastische* Verfahren. In dieser Masterarbeit wird ein *deterministisches* Identifikationsverfahren verwendet, da das Mensch-Fahrrad-System keine stochastischen Informationen beinhaltet. Deterministische Verfahren setzen voraus, dass zusammengehörige Messungen oder Datensätze von Ein- und Ausgangssignalverläufen eines dynamischen Systems gegeben sind. Dabei sind die Eingangssignale definierte Testsignale. Die Ausgangssignalverläufe sind die Reaktionen des Systems auf die Testsignale. Ziel ist es, die Struktur und Parameter eines geeigneten mathematischen Modells so zu finden, dass das statische und dynamische Verhalten des realen Systems möglichst genau beschrieben wird.

Unter deterministischen Verfahren, unterscheidet man wieder zwei Verfahren. Das *empirische* Verfahren, wie z.B. *Schwarze* Verfahren, und das Verfahren der kleinsten Quadrate der kennwertlinearen Fehler. Hier wird das Verfahren der kleinsten Quadrate (*LS-Schätzer*) kurz beschrieben, da es in dieser Arbeit zur Mensch-Modellierung verwendet wird.

Der *LS* (*Least Square*)-*Schätzer* versucht die Fehlerquadratsumme zwischen den Ausgangssignalen von Modell und System zu minimieren, wobei Modell und System durch das gleiche Eingangssignal erregt werden. Als Modellansatz im vorliegenden Fall eignet sich prinzipiell die Z-Übertragungsfunktion

$$G_n(z^{-1}) = \frac{b_0 + b_1 \cdot z^{-1} + \cdots + b_n \cdot z^{-n}}{1 + a_1 \cdot z^{-1} + \cdots + a_n \cdot z^{-n}} = \frac{Y(z^{-1})}{U(z^{-1})} \tag{5.1}$$

Im Zeitbereich lässt sich damit das zeitdiskrete Ausgangssignal $y(k)$ prinzipiell darstellen als

$$y(k) = b_0 \cdot u(k) + \cdots + b_n \cdot u(k-n) - a_1 \cdot y(k-1) - \cdots - a_n \cdot y(k-n) \tag{5.2}$$

Bei dem System, in welchem sich die aktuelle Eingangsgröße *u(k)* nicht unmittelbar auf die Ausgangsgröße *y(k)* auswirkt, wird im Modellansatz auf einen Term $b_0 \cdot u(k)$ verzichtet, d.h. im Modellansatz wird kein Durchgriff vorgesehen. Daher wird jedoch nicht der Ansatz nach Gleichung (5.2), sondern der modifizierte Ansatz

$$y(k) = b_1 \cdot u(k-1) + \cdots + b_n \cdot u(k-n) - a_1 \cdot y(k-1) - \cdots - a_n \cdot y(k-n) \tag{5.3}$$

verwendet. In (5.2) bilden $b_0, \cdots, b_n, a_1, \cdots, a_n$ den Parametervektor $\underline{m}$. n ist die Ordnung des Modellansatzes.

Das *LS-Verfahren* versucht nun, die Fehlerquadratsumme I zu minimieren:

$$I = \sum_{k=1}^{N} \varepsilon^2(k) = \overset{\min}{\underline{m}} \tag{5.4}$$

I ist eine Funktion des verallgemeinerten, kennwertlinearen Fehlers $\varepsilon_v(k)$, der für diesen Modellansatz folgendes Aussehen hat:

$$\varepsilon(k) = \varepsilon_v(k) = y(k) - (u(k),\cdots,u(k-n),-y(k-1),\cdots,-y(k-n)) \cdot \begin{pmatrix} b_0 \\ \vdots \\ b_n \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \tag{5.5}$$

Die zeitdiskreten Ein- und Ausgangsgrößen $u(k),\cdots,u(k-n),-y(k-1),\cdots,-y(k-n)$ werden als Datenvektor $\underline{g}^T(k)$ zusammengefasst. Man geht nun bei der Minimierung wie bei einer skalaren Gleichung vor.

Gegeben seien N Messungen des Ausgangsgröße y. Man erhält hiermit die folgenden N Gleichungen für den verallgemeinerten Ausgangsfehler:

$$\begin{pmatrix} \varepsilon(1) \\ \vdots \\ \varepsilon(N) \end{pmatrix} = \begin{pmatrix} y(1) \\ \vdots \\ y(N) \end{pmatrix} - \begin{pmatrix} \underline{g}^T(1) \\ \vdots \\ \underline{g}^T(N) \end{pmatrix} \cdot \underline{m} \tag{5.6}$$

oder in Matrizenschreibweise

$$\underline{\varepsilon} = \underline{y} - \underline{G} \cdot \underline{m} \tag{5.7}$$

Der Lösungsansatz besteht nun in der Minimierung der Fehlerquadratsumme:

$$J = \underline{\varepsilon}^T \cdot \underline{R} \cdot \underline{\varepsilon} \tag{5.8}$$

Hierin sei $\underline{R}$ eine symmetrische und positiv definite Gewichtungsmatrix. Setzt man (5.7) in (5.8) ein, so erhält man

$$J = \underline{y}^T \underline{R} \cdot \underline{y} - \underline{y}^T \underline{R} \cdot \underline{G} \cdot \underline{m} - \underline{m}^T \underline{G}^T \underline{R} \cdot \underline{y} + \underline{m}^T \underline{G}^T \underline{R} \cdot \underline{G} \cdot \underline{m} \tag{5.9}$$

Das Minimum dieser Funktion kann bestimmt werden, indem man die Funktion nach dem Vektor $\underline{m}$ ableitet und die Ableitung gleich dem Null-Vektor setzt.

$$\frac{dJ(\underline{m})}{d(\underline{m})} = -\underline{G}^T \underline{R}^T \underline{y} - \underline{G}^T \underline{R} \cdot \underline{y} + 2 \cdot \underline{G}^T \underline{R} \cdot \underline{G} \cdot \underline{m} = \underline{0} \tag{5.10}$$

Beachtet man, dass $\underline{R}$ symmetrisch ist, so ergibt sich

$$-2 \cdot \underline{G}^T \underline{R} \cdot \underline{y} + 2 \cdot \underline{G}^T \underline{R} \cdot \underline{G} \cdot \underline{m} = \underline{0} \tag{5.11}$$

Durch Umformung erhält man die Lösung:

$$(\underline{m}) = (\underline{G}^T \cdot \underline{R} \cdot \underline{G})^{-1} \cdot \underline{G}^T \cdot \underline{R} \cdot \underline{y} \tag{5.12}$$

Diese Gleichung stellt prinzipiell den Algorithmus zur direkten Berechnung der Parameter dar. Die direkte Berechnung von $\underline{m}$ entsprechend (5.12) ist eine Offline-Identifikation. Offline deshalb, da die Daten des Systems bereits aufgenommen wurden und dann zur Modellkonstruktion herangezogen werden. Sie hat jedoch den Nachteil, dass bei großem N sehr viel Speicherplatz notwendig ist und bei wachsendem N die Dimension der Matrix ebenfalls wächst. Weiterhin müssen bei der direkten Methode viele Durchläufe gemacht werden, um das Modell iterativ zu verbessern. Daher verwendet man sogenannte rekursive Algorithmen, um diesen Prozess zu beschleunigen. Diese Vorgehensweise realisiert eine Online-Identifikation. Sie kann direkt am System vorgenommen werden, d. h. der Datenvektor $\underline{g}^T(k)$ wird online eingelesen und ständig aktualisiert. Ein weiterer Vorteil des Online-Verfahrens ist die Datenkompression: Die benötigte Datenmenge reduziert sich, da alte Werte nicht mehr benötigt werden, sobald neue hinzukommen. Nachteile dieser Methode bestehen zum einen darin, dass der Modellansatz a priori gewählt werden muss. In dieser Masterarbeit wird das Offline-Verfahren verwendet, weil bei dem Offline-Verfahren mehrere Modellansätze ausprobiert werden können.

Um das Mensch-Modell mit *LS-Verfahren* zu identifizieren, wird zuerst ein Testsignal auf das Mensch-Fahrrad-System geschaltet und das daraus resultierende Herzfrequenzsignal des Fahrers gemessen. Wegen der vielseitigen Abhängigkeit der Herzfrequenz eines Radfahrers während des Trainings von z.B. Trittfrequenz, Trainingsbereich, Fahrradübersetzung usw., wird hier der Stufentest zur Mensch-Modellierung verwendet.

5.1.2 Stufentest

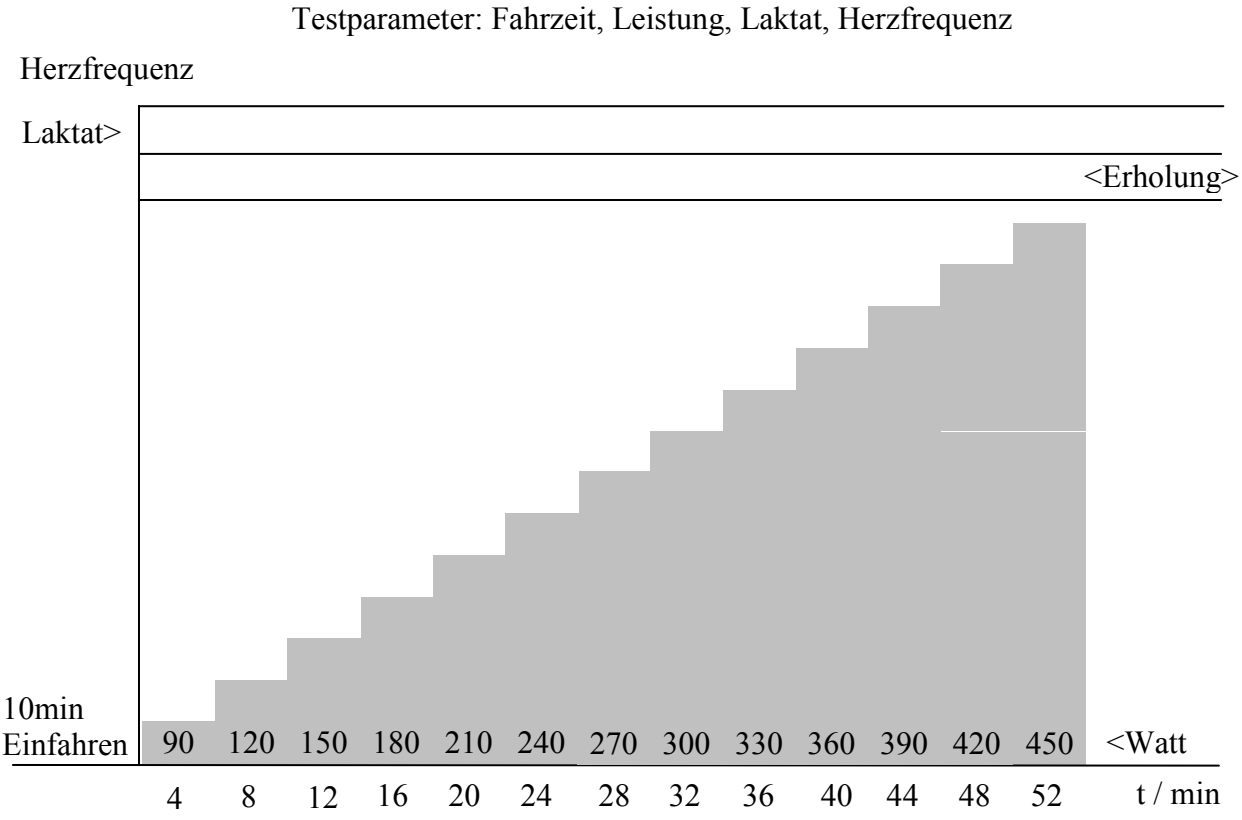


Abbildung 5.1: Testaufbau Stufentest auf dem Fahrradergotrainer

Der Stufentest auf dem Fahrradergotrainer ist die am weitesten verbreitete Testmethode zur Leistungsdiagnose im Radsport. In der Abbildung 5.1 sind der Testaufbau und die Untersuchungsmethodik dargestellt. Die Testperson nimmt auf dem Ergotrainer Platz. Dabei sollte die Sitzposition so eingestellt werden, dass sie der Originalposition auf dem Rennrad entspricht, Pedalsysteme und Schuhe inbegriffen. Nach der 10-minütigen Phase des Warmfahrens mit geringer Intensität beginnt der eigentliche Test mit einer geringen Eingangsleistung von z.B. 90 Watt. In gleichen Zeitrhythmen (3, 4 oder 5 Minuten) erfolgen gleiche Belastungssteigerungen (20, 30, 40 oder 50 Watt). Während des gesamten Stufentests muss eine vorgegebene Trittfrequenz (90 oder 100 U/min) eingehalten werden. Kann der Sportler die vorgegebenen Kriterien der Leistung nicht mehr einhalten, gilt der Test als beendet.

Der Ergotrainer zeichnet alle Testparameter (Leistung, Trittfrequenz und Herzfrequenz) der einzelnen Stufen auf und signalisiert optisch oder akustisch, wenn der Sportler die vorgegebenen Werte nicht mehr erreicht. Während des Tests wird das Laktat auf jeder Stufe gemessen. Zum Standard des Stufentests gehört weiterhin die Registrierung der Sauerstoffaufnahme. Zur Beurteilung der Parameter werden diese absolut angegeben und auf pro Kilogramm Körpergewicht des Sportlers umgerechnet. Weitere Informationen über Stufentest sind in [Lin-00] zu finden.

In unserem Fall werden die Tritt- und Herzfrequenz des Radfahrers sowie die Geschwindigkeit des Rades während des Stufentests gemessen. Die gemessenen Daten und das Leistungstestsignal des Eingangs werden in ein Diagramm übertragen und aufgezeichnet. Aus diesem Diagramm können dann Aussagen über die aerobe und anaerobe Schwelle, den aktuellen Leistungsstand, Stärken und Schwächen sowie Trainingstipps abgeleitet werden. Ein besonderer Vorteil des Stufentests für die Mensch-Modellierung ist die Erkennung der aeroben und anaeroben Schwelle des Radfahrers. Damit wird es möglich sein, einen vernünftigen Belastungsbereich auszuwählen, um ein gutes Mensch-Modell, mit dem das allgemeine Verhalten des menschlichen Herz beschrieben werden kann, zu identifizieren.

5.1.3 Durchführung des Stufentests

Um den Radfahrer zu modellieren, werden viele Stufentests mit unterschiedlichen Testprobanden durchgeführt. In der Abbildung 5.2 ist das Ergebnis eines Stufentests zu sehen. Die Verläufe der Herzfrequenz, der Trittfrequenz, der Geschwindigkeit und der Leistung dieses Tests sind in der Abbildung aufgezeichnet. Dieser Test beginnt mit einer Eingangsleistung von 120 Watt. In einem 4-minütigen Zeitrhythmus erfolgt eine Leistungssteigerung von je 30 Watt. Bevor der Test beginnt, ist der Testfahrer ungefähr zehn Minuten warmgefahren und hat eine Herzfrequenz von 105 S/min. Während des gesamten Tests, fährt der Testfahrer mit einer relativ konstanten Trittfrequenz von 90 U/min. Der gesamte Test dauert ungefähr 35 Minuten und hat insgesamt neun Stufen. Nach acht vollen Stufen und eine Minute in der neunten Stufe ist der Test beendet. Der Testfahrer fährt noch ungefähr fünf Minuten mit geringer Leistung, um sich zu erholen. Aus dem aufgezeichneten Testergebnis ist zu erkennen, dass die aerobe Schwelle des Testfahrers ungefähr bei der Herzfrequenz von 150 S/min (bei der Zeit von ungefähr 1000 Sekunden) liegt. Bis zu diesem Punkt, steigt seine Herzfrequenz nach jeder Belastungssteigerung an und bleibt dann relativ konstant. Ab diesem Punkt, steigt die Herzfrequenz ständig weiter, obwohl keine Belastungssteigerung stattfindet.

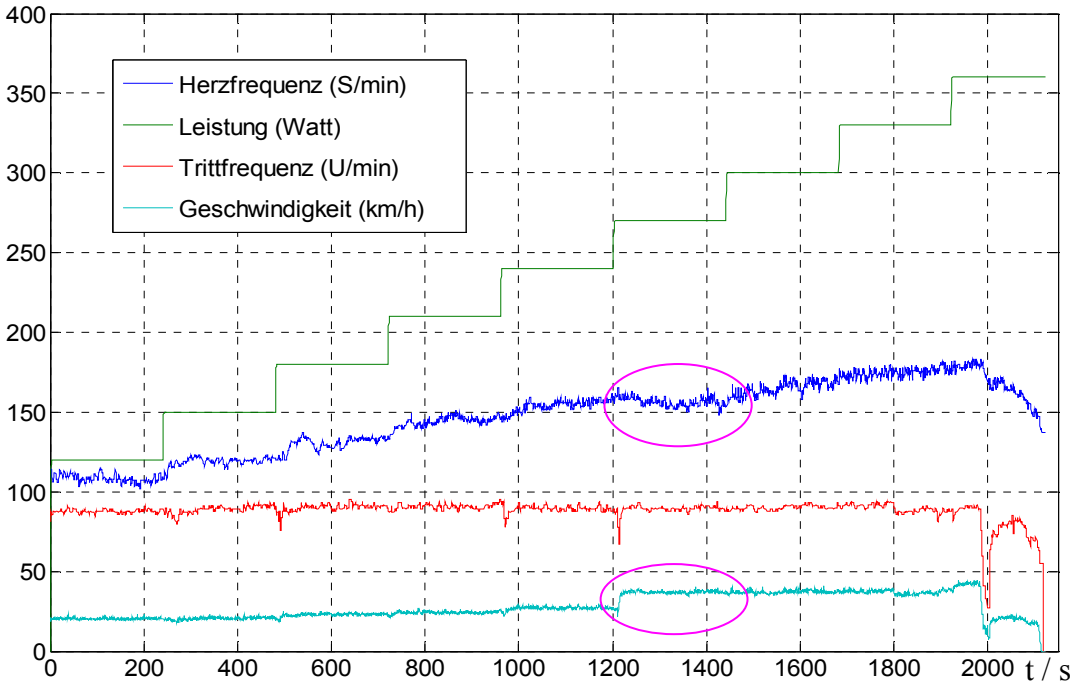


Abbildung 5.2: Stufentestergebnis eines trainierten Testprobanden

An der Abbildung erkennt man, dass der Herzfrequenzverlauf des Radfahrers während des Tests von der Fahrradübersetzung abhängig ist. Aus der Magentamarkierung in der Abbildung ist klar zu erkennen, dass am Anfang der sechsten Stufe beim Zeitpunkt von 1200 Sekunden, der Testfahrer die Übersetzung des Fahrrades um mehrere Stufen einmal umschaltet. Der Testfahrer fährt mit einer relativ konstanten Trittfrequenz, aber ab diesem Zeitpunkt steigt die Geschwindigkeit stark an. Dies führt zu einer leichten Senkung der Herzfrequenz in der Mitte der sechsten Stufe. Es ist also wichtig, dass während des Tests die Übersetzung des Fahrrades möglichst konstant zu halten oder nur ein geringer Übersetzungswechsel erfolgt.

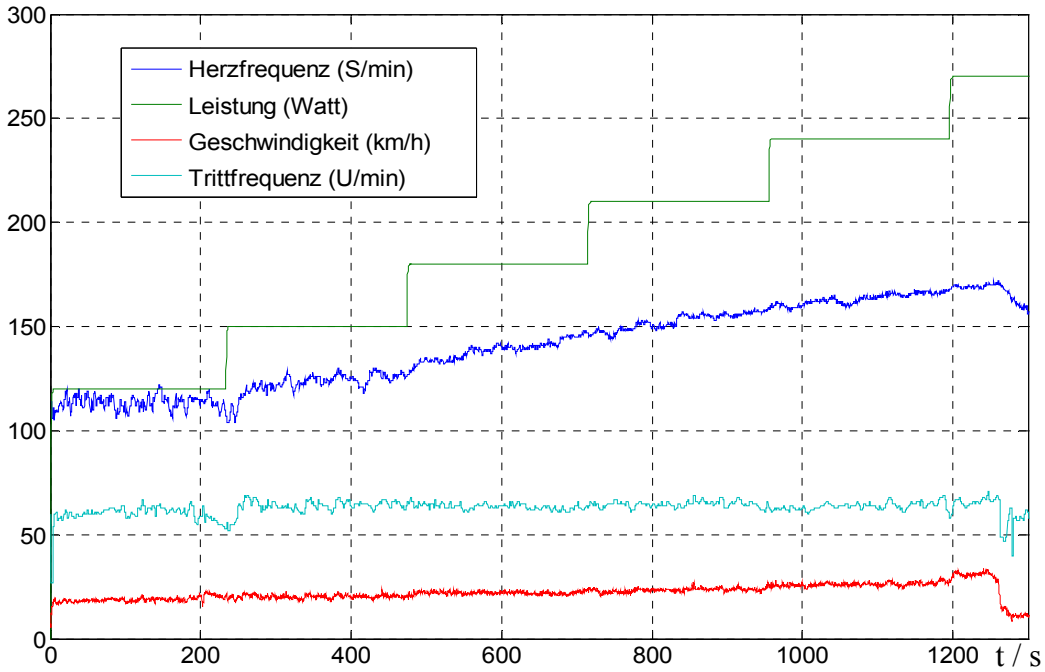


Abbildung 5.3: Stufentestergebnis eines nicht trainierten Testprobanden

In Abbildung 5.3 ist das Testergebnis eines nicht trainierten Testprobanden dargestellt. Dieser Test wurde auch in einem 4-minütigen Zeitrhythmus mit einer Leistungssteigerung von 30 Watt durchgeführt. Der Testfahrer fährt mit den vorgegebenen Testkriterien nur fünf Stufen und hat eine niedrige Trittfrequenz von ungefähr 65 U/min. Nach der ersten Stufe, steigt die Herzfrequenz des Testfahrers ständig an bis zum Testende. Es ist unmöglich aus diesem Herzfrequenzverlauf die Leistungssteigerung zu erkennen. Das bedeutet, dass dieses Testergebnis des nicht trainierten Testprobanden nicht geeignet zur Identifikation des Mensch-Modells ist.

Nach verschiedenen Stufentests mit unterschiedlichen Testprobanden und Testkriterien wurden die folgenden Erkenntnisse für weitere Tests gewonnen:

1. Man braucht gut trainierte Probanden für die Tests; die Testergebnisse von nicht trainierten Probanden sind nicht geeignet zur Identifikation des Mensch-Modells.

2. Während des gesamten Tests sollte möglichst kein oder nur ein geringer Übersetzungswechsel erfolgen.

3. Die Herzfrequenzmessung ist fehlerbehaftet. Durch Verrutschen des Herzfrequenzsensors, treten Messfehler im Herzfrequenzverlauf auf. Außerdem muss der Abstand zwischen dem Sender und Empfänger des Herzfrequenzsensors kleiner als ein Meter sein.

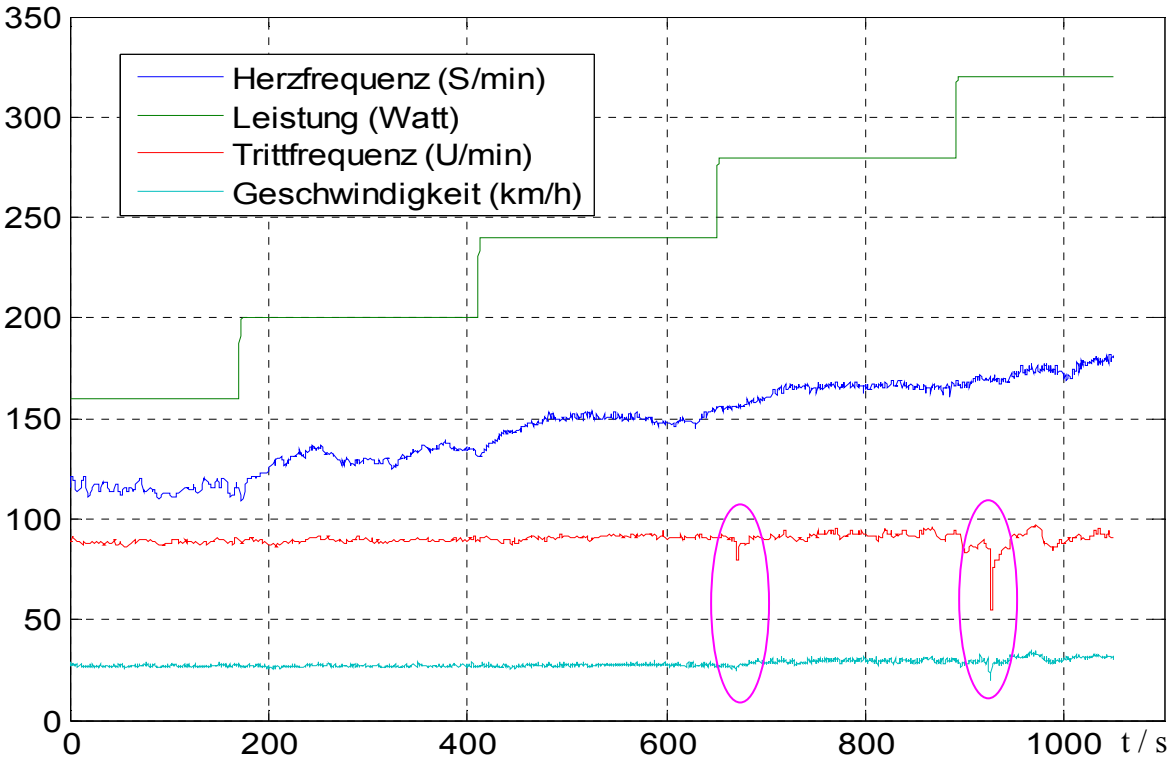


Abbildung 5.4: Stufentestergebnis zur Mensch-Modellierung

Auf Basis dieser Erkenntnisse und mit Berücksichtigung der aeroben Schwelle des Testfahrers, wurden wieder einige Stufentests mit gut trainierten Testprobanden durchgeführt. Das in Abbildung 5.4 dargestellte Ergebnis ist eine Teilaufnahme eines Stufentests, der in einen 4-minütigen Zeitrhythmus mit einer Leistungssteigerung von 40 Watt durchgeführt wurde.

Während des gesamten Tests fährt der Testfahrer mit einer relativ konstanten Trittfrequenz von 90 U/min und hat nur zwei Mal die Übersetzung auf eine andere Stufe umgeschaltet: jeweils am Anfang der Stufe vier und fünf (siehe Magentamarkierung in Abbildung 5.4). Aus diesem Testergebnis ist zu erkennen, dass die Herzfrequenz des Testfahrers nach der Leistungssteigerung langsam ansteigt und dann relativ konstant bleibt. Mit diesem Ergebnis kann die Modellierung des Menschen durchgeführt werden.

5.1.4 Mensch-Modellierung mit *LS-Schätzer*

In diesem Abschnitt wird das Mensch-Modell mit *LS-Schätzer* durch Verwendung des in Abbildung 5.4 dargestellten Testergebnises identifiziert. Als Modellansatz im diesem Fall eignet sich die Gleichung

$$HF(k) = HF_0 + K_1 \cdot P(k) + K_2 \cdot HF(k-1) \tag{5.13}$$

Wobei HF_0 eine Konstante, HF die Herzfrequenz des Menschen, P die Leistung und k der Abtastzeitpunkt ist. Durch Umformung von (5.13) ergibt sich

$$HF(k) = (1, P(k), HF(k-1)) \cdot \begin{pmatrix} HF_0 \\ K_1 \\ K_2 \end{pmatrix} \tag{5.14}$$

Dann hat der verallgemeinerte, kennwertlineare Fehler $\varepsilon_v(k)$ folgendes Aussehen:

$$\varepsilon_v(k) = HF(k) - (1, P(k), HF(k-1)) \cdot \begin{pmatrix} HF_0 \\ K_1 \\ K_2 \end{pmatrix} \tag{5.15}$$

In (5.15), bilden $1, P(k), HF(k-1)$ den $\underline{g}^T(k)$ und $\begin{pmatrix} HF_0 \\ K_1 \\ K_2 \end{pmatrix}$ den $\underline{m}$ Vektor. Durch Anwendung der N Messungen des Stufentests, nach Gleichung (5.6) ergibt sich der verallgemeinerte Ausgangsfehler

$$\begin{pmatrix} \varepsilon(1) \\ \vdots \\ \varepsilon(N) \end{pmatrix} = \begin{pmatrix} y(1) \\ \vdots \\ y(N) \end{pmatrix} - \begin{pmatrix} 1, P(1), HF(0) \\ \vdots \\ 1, P(N), HF(N-1) \end{pmatrix} \cdot \begin{pmatrix} HF_0 \\ K_1 \\ K_2 \end{pmatrix} \tag{5.16}$$

Für *HF(0)* wird hier der erste gemessene Herzfrequenzwert verwendet. Nach Gleichung (5.12) mit $\underline{R}$ gleich $\underline{I}$, der Einheitsmatrix, ergibt sich der Vektor $\underline{m}$

$$\underline{m} = \begin{pmatrix} HF_0 \\ K_1 \\ K_2 \end{pmatrix} = \begin{pmatrix} 0{,}9843 \\ 0{,}0067 \\ 0{,}9823 \end{pmatrix} \tag{5.17}$$

Setzt man (5.17) in (5.13) ein, so erhält man das mathematische Mensch-Modell

$$HF(k) = 0{,}9843 + 0{,}0067 \cdot P(k) + 0{,}9823 \cdot HF(k-1) \tag{5.18}$$

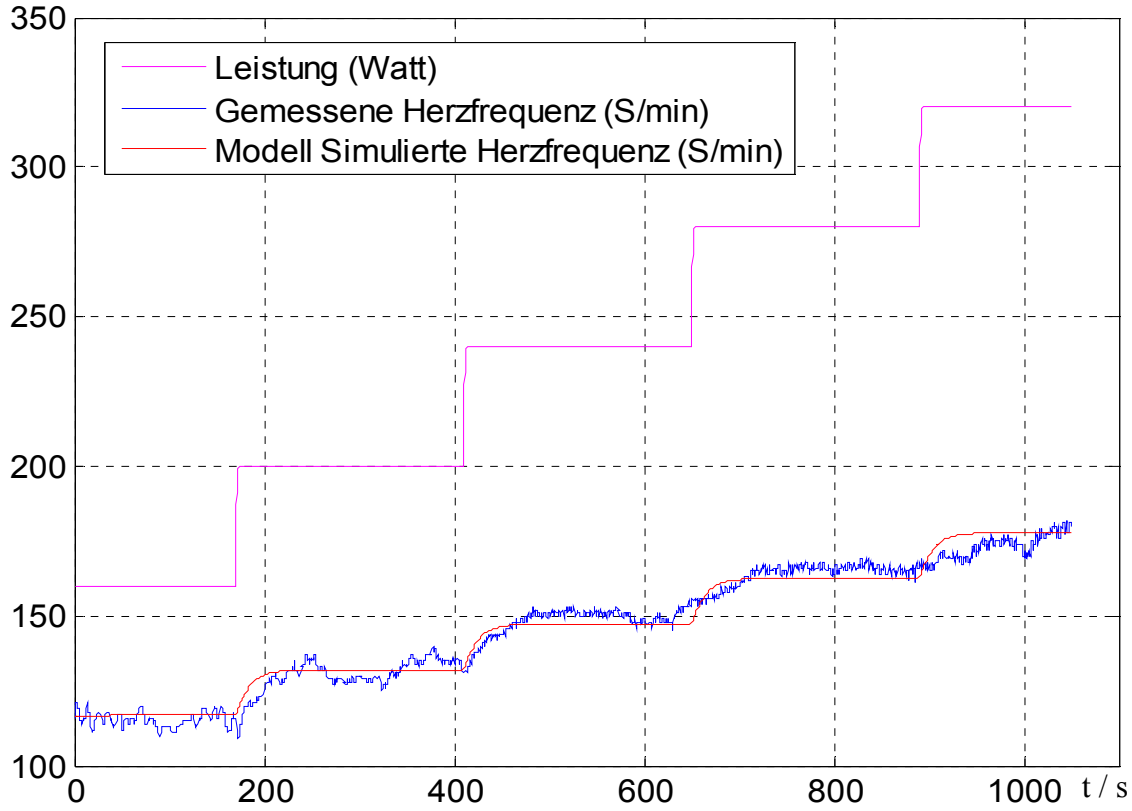


Abbildung 5.5: Messdaten und Modellausgang

Um das identifizierte Menschmodell zu beurteilen, wird das gleiche Testsignal wie beim Stufentest an den Modelleingang angelegt. Das resultierende Herzfrequenzsignal ist zusammen mit der gemessenen Herzfrequenz in der Abbildung 5.5 aufgezeichnet. Aus der Aufzeichnung ist zu erkennen, dass das vom Modell resultierende Signal die gemessene Herzfrequenz gut simulieren kann.

Für das modellbasierte Reglerentwurfsverfahren, soll das Modell als Übertragungsfunktion ausgedrückt werden. Aus Gleichung (5.13) erhält man

$$HF(k-1) = HF_0 + K_1 \cdot P(k-1) + K_2 \cdot HF(k-2) \tag{5.19}$$

Zieht man (5.19) von (5.13) ab, ergibt sich

$$HF(k) - HF(k-1) = K_1 \cdot (P(k) - P(k-1)) + K_2 \cdot (HF(k-1) - HF(k-2)) \tag{5.20}$$

Durch Umformung von (5.20) erhält man

$$HF(k) - (1 + K_2) \cdot HF(k-1) + K_2 \cdot HF(k-2) = K_1 \cdot P(k) - K_1 \cdot P(k-1) \tag{5.21}$$

Durch Transformation von (5.21) in den Z-Bereich, ergibt sich

$$[1-(1+K_2)\cdot z^{-1}+K_2\cdot z^{-2}]\cdot HF(z^{-1})=K_1\cdot(1-z^{-1})\cdot P(z^{-1}) \tag{5.22}$$

Durch Umformung von (5.22) erhält man die Z-Übertragungsfunktion des Menschmodells

$$G_S(z^{-1})=\frac{HF(z^{-1})}{P(z^{-1})}=\frac{K_1\cdot(1-z^{-1})}{(1-z^{-1})\cdot(1-K_2\cdot z^{-1})} \tag{5.23}$$

Das Modell hat die Ordnung 1 und der Term $(1-z^{-1})$ wird im Zähler und Nenner von (5.23) gekürzt. Setz man die Werte von K_1 und K_2 in (5.23) ein, erhält man das Mensch-Modell

$$G_S(z^{-1})=\frac{HF(z^{-1})}{P(z^{-1})}=\frac{0{,}0067}{1-0{,}9823z^{-1}} \tag{5.24}$$

Durch Verwendung dieses Mensch-Modells als Streckenmodell, kann der Reglerentwurf mit dem modellbasierten Verfahren durchgeführt werden.

5.2 Regler-Entwurf

Es gibt sehr viele, teils aufwändige Möglichkeiten, einen Regleralgorithmus zu entwerfen und zu implementieren. Die lineare Standardregler P-, PI-, PD- und PID-Regler (wobei die ersten drei ein Sonderfall des PID sind) sind sehr gut erforscht, relativ einfach zu entwerfen und zu implementieren. Aufgrund der guten Regeleigenschaften eines PI-Reglers und einer guten Realisierbarkeit mit den vorhandenen Hardwarekomponenten, wird ein PI-Regler in dieser Masterarbeit als Herzfrequenzregler verwendet.

Es existieren verschiedene Möglichkeiten, einen PI-Regler zu entwerfen. Am einfachsten geht es mit den Einstellregeln für lineare Standardregler. Diese Verfahren haben eine lange Tradition und ermöglichen eine direkte Inbetriebnahme an der realen Regelstrecke. Als Beispiele seien hier das Verfahren nach Ziegler/Nichols und das nach Chien/Hrones/Reswick genannt. Komplizierte Verfahren benötigen ein tieferes Verständnis der Regelungstheorie und liefern auf die jeweilige Strecke sehr gut angepasste Regler. Beispielsweise kann der Regler mit Hilfe der WOK oder durch das FKL-Verfahren entworfen werden.

In dieser Arbeit werden diese Verfahren mit Hilfe von WOK und FKL zum Entwurf des Herzfrequenzreglers verwendet. Auf das genaue Prinzip und die Grundlagen dieser Verfahren wird dabei verzichtet, da diese bei der Masterarbeit nicht von primärem Interesse sind und als bekannt vorausgesetzt werden dürfen. Alle Informationen über Regelungstechnik können [Föl-85], [Föl-93] und der Vorlesungen “Regelungstechnik I“ und “Regelungstechnik II“ vom Professor S. Liu entnommen werden.

Vor Beginn des Herzfrequenzreglerentwurfs wird noch kurz auf die Spezifikationen des Regelungssystems und die Problematik des Windup-Effekts des Integrators eingegangen.

5.2.1 Spezifikationen des Regelungssystems

Ein Regelkreis soll nach Fertigstellung bestimmte Aufgaben erfüllen. Üblicherweise werden heutige Regler in digitaler Form realisiert, das Regelgesetz liegt also am Ende als Software vor. Jeder Softwareentwurf startet mit der informellen Spezifikation, d.h. Anforderungen die an die Software (bzw. hier den Regelkreis) gestellt werden, sind allgemeinverständlich, ohne an eine bestimmte Syntax gebunden zu sein, formuliert.

Allgemeine Spezifikationen, wie sie bei jedem Regelkreis anzutreffen sind (aus [Lit-05]):

RS1: Ein Sollwert oder mehrere Sollwerte sind einzuregeln.

RS2: Störungen, die auf die Regelstrecke einwirken sind auszuregeln.

RS3: Der Regelkreis muss stabil sein.

RS4: Das Regelsystem muss robust sein.

RS1 – RS4 sind so allgemein, dass sie für jedes Regelungssystem gelten. Je nach Anwendungsfall gibt es auch optionale Spezifikationsteile, die in verschiedenen Kombinationen anzutreffen sind.

RS5: Das Regelungssystem soll stationär genau sein.

RS6: Die Überschwingweite ü soll begrenzt sein.

RS7: Die Führungssprungantwort soll aperiodisch einschwingen.

RS8: Der Ausregelfehler a soll begrenzt werden.

RS9: Obere Grenzen für Anregelzeit T_{an} bzw. Anstiegszeit T_R und für die Ausregelzeit T_{aus} sollen nicht überschritten werden.

RS10: Die Stellgröße u(t) sei gemäß $u_{min} \leq u(t) \leq u_{max}$ beschränkt.

5.2.2 Problematik des Windup-Effekts

An dieser Stelle wird auf die Ursache und die Wirkung des Windup-Effekts des I-Reglers (Integrator) und einen möglichen Ansatz einer Anti-Windup-Maßnahme eingegangen. Zu Beginn wird die Beschränkung der Stellgröße beim praktischen Regelungssystem kurz vorgestellt.

5.2.2.1 Beschränkung der Stellgröße

Bei praktisch allen regelungstechnischen Anwendungen darf sich die Stellgröße u(t) nur in bestimmten Grenzen bewegen. Es kann dafür 2 Gründe geben:

- Leistung des Stellgliedes ist beschränkt
  [z.B. kann ein Ventil max. 100% geöffnet sein und hat damit einen begrenzten maximalen Durchfluss; auch ein Elektromotor hat eine maximale Leistung].

- Strecke ist nicht beliebig belastbar.

Meist sind daher harte Stellgrößenbeschränkungen zu berücksichtigen, was bedeutet, dass der Absolutbetrag der Stellgröße sich in bestimmten Grenzen bewegen muss.

$$|u(t)| \le u_{max} \quad \forall t \tag{5.25}$$

Es gibt zwar Ansätze diese vorhandenen Beschränkungen beim Reglerentwurf zu berücksichtigen, jedoch ist eine gezielte Einhaltung nur schwer möglich. Nutzt man den zur Verfügung stehenden Stellbereich nur schlecht aus, führt dies zu einem unnötig trägen Regelverhalten. Auf der anderen Seite bergen länger andauernde Stellbereichs-Überschreitungen die Gefahr unerwünschten dynamischen Verhaltens.

Ein nahe liegender Lösungsansatz, um diese Stellgrößenbeschränkung im Regelkreis zu berücksichtigen, ist der Einsatz eines "Begrenzers" (engl.: Limiter).

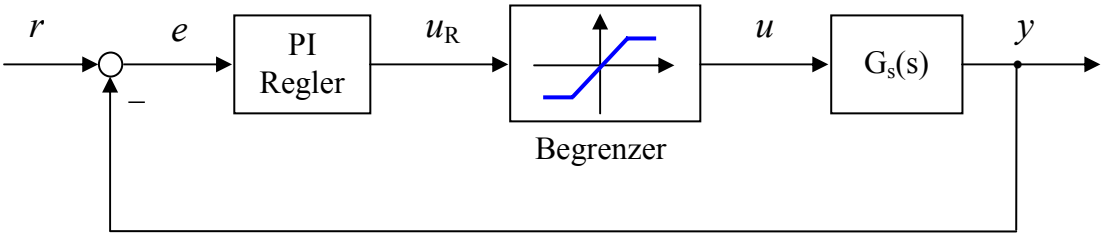


Abbildung 5.6: Berücksichtigung der Stellgrößenbeschränkung durch einen Begrenzer

Die Wirkung des Begrenzers ist mit der einer Sättigungskennlinie vergleichbar:

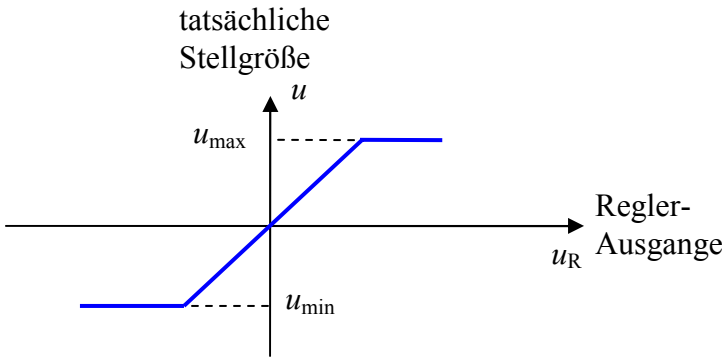


Abbildung 5.7: Typische Begrenzungskennlinie

Bei einer Überschreitung des zulässigen Stellgrößenbereichs wird also die nichtlineare Sättigungskennlinie wirksam. Allerdings birgt dieser Lösungsansatz auch Gefahren: Das dynamische Verhalten des (jetzt) nichtlinearen Regelkreises verschlechtert sich im Allgemeinen – es kann sogar zur Instabilität kommen.

Theoretisch müsste die Stabilität nun mit geeigneten Stabilitätskriterien für nichtlineare Regelkreise untersucht werden. Da dies aber sehr aufwändig sein kann, behilft man sich der Einfachheit halber damit, das Regelverhalten nach der Einführung solcher Begrenzer in Simulationen zu untersuchen und mit dem der zuvor als stabil nachgewiesenen Strecke zu vergleichen. Dies ist zwar kein mathematisch korrekter Nachweis, hilft aber dabei, das neue Stabilitätsverhalten abschätzen zu können.

5.2.2.2 Windup-Effekt

Die Ursache des Windup-Effekts liegt beim I-Anteil des Reglers, da der lang anhaltende Regelfehler (z.B. durch physikalische Begrenzung der Stellgröße) am Integratoreingang zu (unbegrenzt) hohem bzw. niedrigem Wert des Integratorausgangs führt. Der eingeschränkte Stellbereich hat eine schwerwiegende Folge für den I-Anteil des Reglers. Nehmen wir an, es stehe über längere Zeit eine positive Regelabweichung an, wobei gleichzeitig die Stellgröße über diese Zeit $u_{\max}$ beträgt, sich also in der oberen Begrenzung aufhält. Obwohl die tatsächliche Stellgröße nicht mehr erhöht werden kann, wird der I-Anteil während dieser Zeit stets weiter wachsen, je nach Zeitdauer der positiven Regelabweichung bis auf sehr hohe Werte. Hat der I-Anteil erst mal solche Werte erreicht, kann er nur durch Abwärtsintegration wieder verringert werden. Dazu ist eine negative Regelabweichung notwendig. Die tatsächliche Stellgröße verharrt jedoch solange an der oberen Grenze, bis die Summe aus P- und I-Anteil den Wert $u_{\max}$ unterschreitet. Dies führt zu einem ungünstigen und unerwünschten Verhalten des Reglers. Das unnötige Anwachsen der Reglerausgangsgröße bezeichnet man als Windup-Effekt ("Aufwickeln des Integrierers") und führt zu großem Überschwingen.

5.2.2.3 Anti-Windup-Maßnahme

Das Ziel von Anti-Windup-Maßnahmen ist es, dieser unnötigen Integration entgegenzuwirken, wenn die Stellgröße $u(t)$ am oberen Anschlag ist, d.h. wenn die Stellgrößenbeschränkung wirksam wird. Es existieren eine ganze Reihe unterschiedlicher Anti-Windup-Mechanismen. Sie resultieren alle darin, ein Weglaufen des I-Anteils über bestimmte Grenzen zu vermeiden, z.B. die Anti-Windup-Hold und dynamische Anti-Windup-Reset Mechanismen. In dieser Masterarbeit wird der Regler in SIMULINK implementiert, dafür wird folgende Anti-Windup-Maßnahme verwendet.

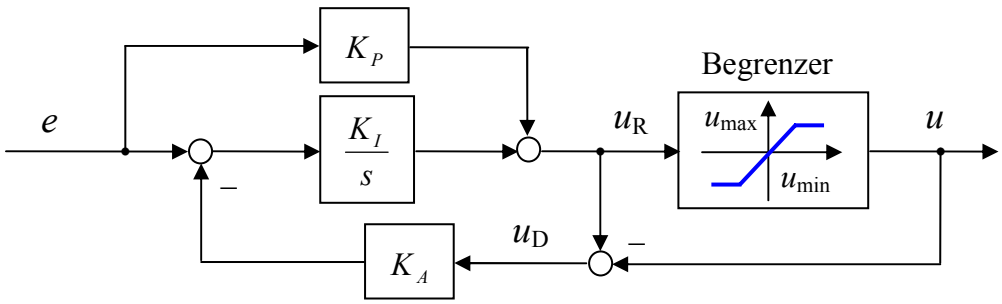


Abbildung 5.8: Mögliche Anti-Windup-Maßnahme

Der PI-Regler wird hier in additiver Form dargestellt. Die Idee dieser Anti-Windup-Maßnahme ist, dass der Reglerausgang u_R ständig mit der tatsächlichen Stellgröße u verglichen und die Differenz mal einen Faktor K_A am Eingang des Integrators zurückgekoppelt wird. Der Faktor K_A ist so zu bestimmen, dass die Stellgröße $u(t)$ am oberen Anschlag gehalten wird, wenn der Begrenzer wirksam wird. Wenn u_R sich im zulässigen Bereich bewegt, d.h. der Begrenzer nicht wirksam wird, nämlich $u_R = u$ und $u_D = 0$, dann ist die Rückkopplung wirkungslos. Um ein besseres Regelverhalten zu erlangen, werden die Begrenzungswerte der Stellgröße u_{max} und u_{min} während des Vorgangs der Regelung dynamisch eingestellt. Dies wird im Abschnitt 6.3 *Blockschaltbild des gesamten Regelungssystems* noch erklärt.

5.2.3 Entwurf des Herzfrequenzreglers

Der Reglerentwurfsprozess umfasst allgemein folgende Schritte:

- Festlegung einer Reglerstruktur bzw. einer geeigneten Reglerklasse.
  → der Regler muss prinzipiell in der Lage sein, die Spezifikationen zu erfüllen.

- Die Reglerklasse enthält freie Parameter. Diese sind so zu bestimmen, dass auch alle zahlenmäßigen Anforderungen des Spezifikationsteils erfüllt werden.
  → man erhält so ein oder mehrere mögliche Reglerindividuen.

- Durch die Validierung wird derjenige der obigen Regler ausgewählt, der die Spezifikationen am besten erfüllt.

Für die Herzfrequenzregelung ist hier die Reglerstruktur schon festgelegt, nämlich einen PI-Regler in additiver Form

$$G_{PI}(s) = K_P + \frac{K_I}{s} \tag{5.26}$$

Um K_P und K_I unter den in der Aufgabenstellung genannten Vorgaben zu bestimmen, sind außerdem die im Abschnitt 5.2.1 aufgelisteten allgemeinen Spezifikationen RS1 bis RS4 noch die unten aufgeführten zusätzlichen Spezifikationen zu erfüllen.

- Das Regelungssystem soll stationär genau sein.

- Die Überschwingweite $\ddot{u}$ soll gemäß $\ddot{u}_{\max} < 5\%$ begrenzt sein.

- Die Stellgröße $u(t)$ sei gemäß $u_{\min} \le u(t) \le u_{\max}$ beschränkt. $u_{\max}$ und $u_{\min}$ werden während der Regelung gemäß der Führungsgröße dynamisch bestimmt.

- Die Regelung soll schnell sein, aber die Anstiegszeit T_R soll aus dem menschlichen Herzfrequenzverlauf bestimmt werden.

- Die Regelung soll gut gedämpft sein, d.h. die Dämpfung sollte zwischen 0,6 und 0,8 liegen.

- Der Regelkreis soll eine Phasenreserve φ_R von 50° bis 70° haben.

Nach den obigen Spezifikationen, werden die Reglerparameter K_P und K_I auf der Basis des in (5.24) dargestellten Mensch-Modells bestimmt. Durch Verwendung der MATLAB Toolbox SISOTOOL sind folgende vier Reglerparameter-Paare (K_P, K_I) bestimmt worden.

	K_P	K_I	φ_R (°)	d	T_R (s)	$\ddot{u}_{max}$
1	0,429	0,134	66,8	0,69	34,2	4,99%
2	0,8	0,16	69,2	0,71	30,8	4,51%
3	1,248	0,206	70	0,7	25,4	5%
4	1,724	0,25	71,6	0,72	22	4,97%

Tabelle 5.1: Reglerparameter nach unterschiedlichen zahlenmäßigen Spezifikationen

Durch die Validierung der obigen Reglerindividuen am Modell und am Indoor-Demonstrator, wird von den obigen Reglern der Regler Nummer 1 ausgewählt, da dieser die Spezifikationen am besten erfüllt.

$$G_{PI}(s) = 0{,}429 + \frac{0{,}134}{s} \tag{5.28}$$

Dieser Regler hat einen Verstärkungsfaktor von 0,134 und eine Nullstelle in -0,312. Die WOK des geschlossenen und die FKL des offenen Regelkreises mit diesem Regler sind in Abbildung 5.9 dargestellt. Aus der Abbildung ist zu sehen, dass das Regelungssystem eine Dämpfung von 0,69 und eine Phasenreserve von 66,8° hat. In der Umgebung der Durchtrittsfrequenz ω_D = 0,0437 rad/sec hat die Betragskennlinie ein Gefälle von 20 dB/Dekade.

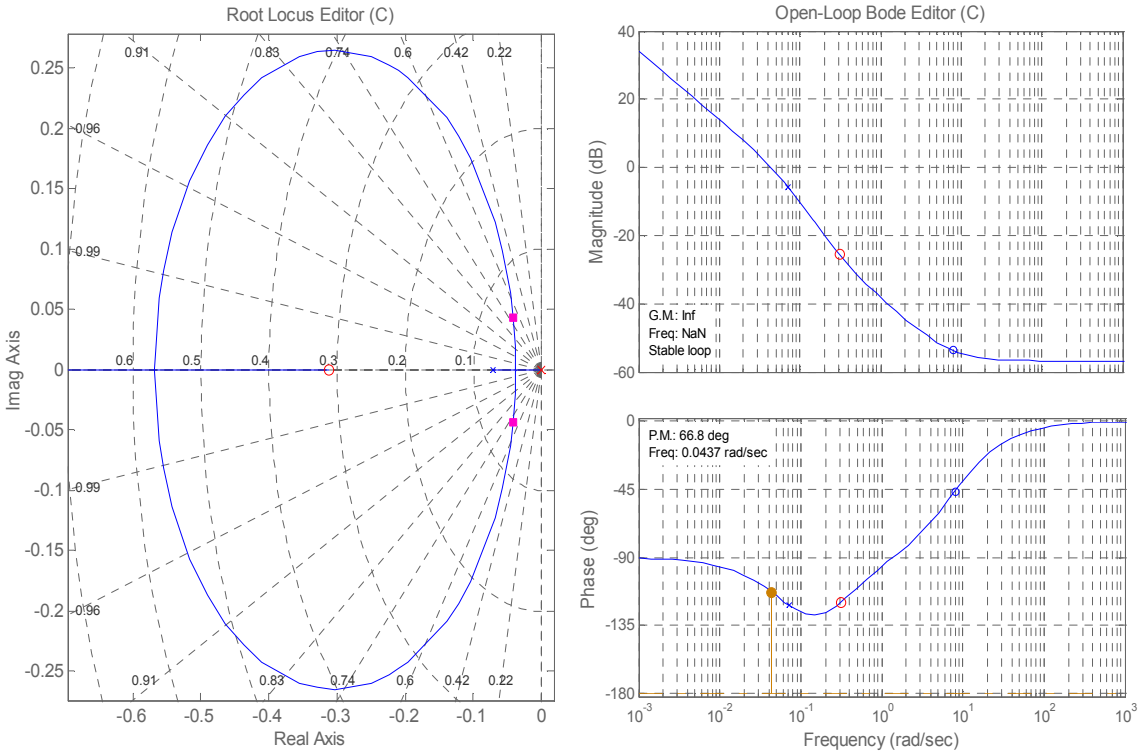


Abbildung 5.9: WOK des geschlossenen Regelkreises und FKL des offenen Regelkreises

Die Sprungantwort der Ausgangsgröße in Abbildung 5.10 zeigt, dass sie eine Anstiegszeit von 34,2 Sekunden und die maximale Überschwingweite von 4,99% hat.

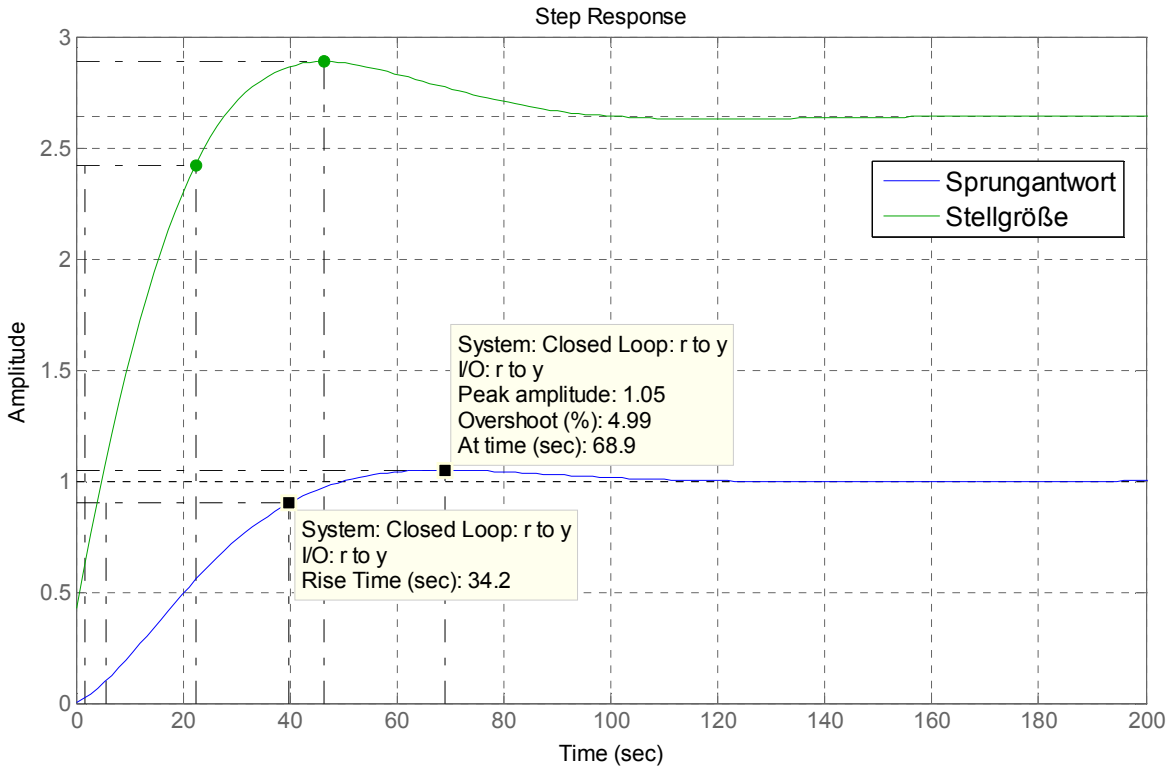


Abbildung 5.10: Sprungantwort der Ausgangs- und der Stellgröße

Aus der WOK und FKL des entworfenen Reglers ist es ersichtlich, dass der Regler die angeforderte Spezifikationen erfüllt. Das Validierungsergebnis anhand des Indoor-Demonstrators wird später im Kapitel 7 *Inbetriebnahme* vorgestellt.

5.3 Zusammenfassung

In den letzten Abschnitten wurde aus den Vorgaben der Aufgabenstellung ein PI-Relger als Herzfrequenzregler für den Indoor-Demonstrator entworfen. Dabei wurden zuerst die Grundlagen der mathematischen Modellierung erläutert. Dann wurde der Stufentest vorgestellt und mit unterschiedlichen Testprobanden durchgeführt. Schließlich wurde aus dem Ergebnis des Stufentests ein lineares mathematisches Modell für das Mensch-Fahrrad-System zum Reglerentwurf identifiziert. Danach wurden die Spezifikationen eines Regelungssystems und die Problematik des Windup-Effekts erklärt. Im Anschluss daran wurden mehrere mögliche Reglerindividuen aus den geforderten Spezifikationen und dem Mensch-Modell ermittelt. Zum Schluss wurde durch Validierung am Modell und am Indoor-Demonstrator derjenige der entworfenen Regler ausgewählt, der die Spezifikationen am besten erfüllt. Dabei ist allerdings zu beachten, dass das Mensch-Modell von individuellen Testfahrern abhängt. Das heißt, dass die Reglerparameter durch Validierung an mehreren verschiedenen Testprobanden noch besser angepasst werden können.

6 Software-Entwurf und Realisierung

In diesem Kapitel wird der Entwurf und die Realisierung der Software für die in den vorherigen Abschnitten entworfenen Konzepte und Regler vorgestellt. Dabei sollen die zuvor entwickelten Ansätze möglichst ohne Veränderungen von der abstrakten Beschreibung in diskrete Softwarestrukturen umgesetzt werden. Allerdings wird hier das eigentliche Programmieren nicht beschrieben, da dies sehr stark von der verwendeten Sprache und dem eingesetzten Compiler abhängig ist.

Zu Beginn wird die verwendete Entwicklungsumgebung und die Systemsoftwarekomponenten kurz vorgestellt. Daran schließt sich der Programmablauf der einzelnen Softwareteile an. Zum Schluss dieses Kaptitels wird das gesamte Regelungssystem mit Anti-Windup-Maßnahme in SIMULINK vorgestellt.

6.1 Entwicklungsumgebung

Wie bereits im vorherigen Abschnitt beschrieben, unterteilt sich die Software dieser Arbeit in zwei Bereiche: den Bereich des Mikroprozessors ATmega128L und den Bereich der Regler in MATLAB/SIMULINK. Im Folgenden werden die Entwicklungsumgebung für den ATmega128L und die Regler sowie die verwendeten Systemsoftwarekomponenten von MATLAB/SIMULINK vorgestellt. Als Erstes wird auf den C-Compiler ICCAVR [Ima-00] der Firma Imagecraft und das AVR Studio der Firma ATmel für das Mikroprozessorprogramm eingegangen.

6.1.1 ICC AVR & AVR Studio

Der C-Compiler ICC AVR der Firma Imagecraft wird als Grundlage zur Programmierung des Mikroprozessors ATmega128L verwendet. Außerdem dient er zur Verbindung mit dem Programmieradapter AVR-ISP (In System Programmer) [Atm-01] für die Übertragung des Programms in den Prozessor. Der C-Compiler ICC AVR stellt eine Reihe von Makros bereit, welche die hardwarenahe, oft bitorientierte Programmierung in C vereinfachen. Ein Programm kann in einem Projekt organisiert werden und aus mehreren Dateien bestehen. Als Ausgabedateien stehen nach der Übersetzung die Dateien mit den Endungen *.cof und *.hex zur Verfügung. Die *.hex Datei enthält das fertige, in Maschinencode übersetzte Programm, und kann mit jedem beliebigen Programmieradapter direkt in den Prozessor geschrieben werden. Zum Beispiel kann man eine *.hex Datei in den Prozessor mit dem AVR-ISP STK500 übertragen. Nachdem die *.hex Datei erstellt wurde, ist der Eintrag “In System Programmer“ im Tools-Menü auszuwählen. Daraufhin öffnet sich ein weiteres Fenster mit dessen Dialogen das Programm in den Prozessor geschrieben oder von dort ausgelesen werden kann. Eine genauere Beschreibung ist in [Atm-01] zu finden. Für den C-Compiler, soll nur noch darauf hingewiesen werden, dass es unbedingt notwendig ist, den richtigen Prozessortyp, nach

dem Anlegen eines neuen Projektes in dem Option-Menü einzustellen. Die genaue Beschreibung des Compilers ist in [Ima-00] zu finden.

Die Datei mit der Endung *.cof enthält neben dem Inhalt der *.hex Datei weiterhin Debuginformationen, welche es ermöglichen das Programm mit Hilfe des AVR-Studios [Web-01] zu simulieren. Das übersetzte Programm kann daraufhin mit dem AVR-Studio simuliert oder aber in den Prozessor eingespielt werden. Hierzu ist zuerst die *.cof Datei im AVR-Studio zu öffnen. Daraufhin wird das Programm im C-Code dargestellt und kann über die entsprechenden Befehle im Debug-Menü simuliert werden. Auf der linken Seite des Bildschirms stehen dabei alle Informationen über den Mikroprozessor, wie Pinbelegungen oder Zählerstände, zur Verfügung. Für eine genauere Beschreibung wird auf die sehr ausführliche Onlinehilfe des Programms verwiesen. Wenn das Programm nicht simuliert, sondern in den Prozessor übertragen werden soll, so ist nachdem die *.cof oder *.hex Datei geöffnet wurde, der Eintrag "AVR-Prog..." im Tool-Menü auszuwählen. Daraufhin öffnet sich wie in ICC AVR ein weiteres Fenster mit dessen Dialogen das Programm in den Prozessor geschrieben oder von dort ausgelesen werden kann. Für eine genauere Beschreibung wird hier ebenfalls auf die Onlinehilfe verwiesen.

6.1.2 MATLAB/SIMULINK

Ein relativ großer Anteil dieser Arbeit wird in MATLAB/SIMULINK implementiert. Datenaustausch zwischen Mikroprozessor und PC, Darstellung der Trainingszustandmeldungen auf dem Display und Leistungsregelung bzw. -steuerung werden in MATLAB *s-Funktion* in Form von level-1 *m-Files* implementiert. Zur Erstellung der Benutzerschnittstelle wird die MATLAB-GUI-Entwicklungsumgebung GUIDE verwendet, welche es ermöglicht, graphisch und objektorientiert Benutzeroberflächen für MATLAB *m-Files* oder SIMULINK-Modelle zu erstellen. Die Herzfrequenzregelung und die Anti-Windup-Maßnahme werden mit Hilfe von *s-Fuktion* in SIMULINK Simulationsblocks realisiert. Folgende Software wurde in dieser Masterarbeit für die Programme auf dem PC verwendet:

- MATLAB/GUIDE Version 7.0.1, Release 14
- SIMULINK Version 6.1, Release 14

Aufgrund der hohen Anzahl der verwendeten Softwarekomponenten ist es nicht möglich, hier sämtliche in dieser Arbeit genutzten Funktionalitäten der einzelnen Programme aufzulisten. Stattdessen werden in dem folgenden Abschnitt einige der am häufigsten verwendeten und einige besondere Funktionen aufgelistet. Die komplette Funktionalität der angegebenen Software kann in den entsprechenden Handbüchern nachgelesen werden.

6.1.2.1 MATLAB und MATLAB GUIDE Komponenten

MATLAB (***mat****rix* ***lab****oratory*) ist eine interaktive, sehr leistungsfähige und weit verbreitete Software zur numerischen Berechnung von Problemen aus dem Bereich der Ingenieurwissenschaften. Dazu zählen Probleme auf dem Gebiet der linearen Algebra, der numerischen Analysis, der Simulation, der Visualisierung der Lösung usw. Der Benutzer verfügt dabei

über einen Satz von speziellen Befehlen, mit denen z.B. direkt Matrizenoperationen oder numerische Integrationen von Funktionen durchgeführt werden können, ohne die dazu notwendigen Routinen selbst programmieren zu müssen. Die Befehle werden dabei interpretiert, d. h. MATLAB nutzt eine Interpreter-Sprache.

MATLAB bietet zwei prinzipielle Arbeitsweisen an. Dieses ist zum einen die interaktive Arbeitsweise, bei der ein Anwender ähnlich wie bei einem Taschenrechner hintereinander seine Befehle über die sogenannte "*Command-Window*" eingibt. Das jeweilige Ergebnis wird direkt auf dem Bildschirm angezeigt. Die zweite Möglichkeit ist das Abspeichern einer Folge von Befehlen in Form eines sogenannten *m-Files*, welches dann komplett abgearbeitet wird. Diese Arbeitsweise entspricht derjenigen, die vom Umgang mit den gewöhnlichen Programmiersprachen her bekannt ist.

MATLAB bietet für spezielle Bereiche der Ingenieurwissenschaften, z.B. der Modellierung und Simulation, sogenannte Toolboxen an. Diese Toolboxen bestehen aus vorgefertigten *m-Files*, die dann für die entsprechende Problemklasse zusätzlich zu den Standard-MATLAB-Befehlen problemspezifische Befehle anbieten. Eine spezielle Toolbox ist SIMULINK. Der Benutzer hat damit die Möglichkeit, graphisch mathematische Modelle aus vorgefertigten Blöcken zu erstellen und diese dann zu simulieren.

An dieser Stelle sollen lediglich die in dieser Arbeit verwendeten Funktionalitäten und Toolboxen von MATLAB aufgelistet werden. Sämtliche Funktionen können [Web-05] entnommen werden. In diesem Abschnitt werden die für die gegebene Problematik entscheidenden Elemente betrachtet. Diese lassen sich auf ein Dutzend begrenzen und werden in Tabelle 6.1 dargestellt.

Notation im Quelltext	Erklärung
load(Datei.mat) var1,var2…	Dieser Befehl lädt die Variable var1, var2 … vom MATLAB-Worksapce-Datei "Datei.mat"
save(Datei.mat, var1,var2…)	Dieser Befehl speichert die Variable var1, var2 … in das MATLAB-Workspace-Datei "Datei.mat"
S=serial('PORT','E1','E2'…)	Dieser Befehl erstellt ein mit 'PORT' verbundenes serielles Port-Objekt heißt S mit Eigenschaften E1, E2 ….
num2str(Nummer,Stelle)	Dieser Befehl konvertiert die in "Nummer" gegebene Zahl in einen String mit in "Stelle" zugewiesenen Stellen
strcmp('String1','String2')	Dieser Befehl vergleicht die in "string1" und "string2" gegebenen Zeichenketten. Sind diese identisch, wird 1 zurück gegeben, sonst 0
YI=interp1(X, Y, XI)	Dieser Befehl erzeugt Data YI durch Interpolation einer 1-dimensionaler Funktion Y von X an der Stelle, wo XI liegt
ZI=interp2(X, Y, Z, XI, YI)	Dieser Befehl erzeugt Data ZI durch Interpolation einer 2-dimensionaler Funktion Z von X und Y an der Stelle, die XI und YI zugewiesen ist
SISOTOOL(Gs)	Es ist ein Reglerentwurfswerkzeug für SISO-Systeme mit interaktiver graphischer Benutzerschnittstelle, gestützt von Ortskurve, WOK und FKL des zu entwerfenden Regelkreises (Gs)

Tabelle 6.1: Häufig verwendete MATLAB Komponenten

Zum einfachen Erstellen einer GUI wird in MATLAB die Toolbox GUIDE verwendet. Diese wird durch Eingabe des Befehls *guide* in das *Command-Window* gestartet. Nach Öffnen einer bereits existierenden GUI oder Erstellen einer neuen öffnet sich der GUIDE-Layout-Editor (siehe Abbildung 6.1). Dieser ermöglicht es dem Benutzer, die Schnittstellenelemente wie *Push Button*, *Slider*, *Edit Text*, *Static Text* usw. in das Fenster zu ziehen und dort anzuordnen.

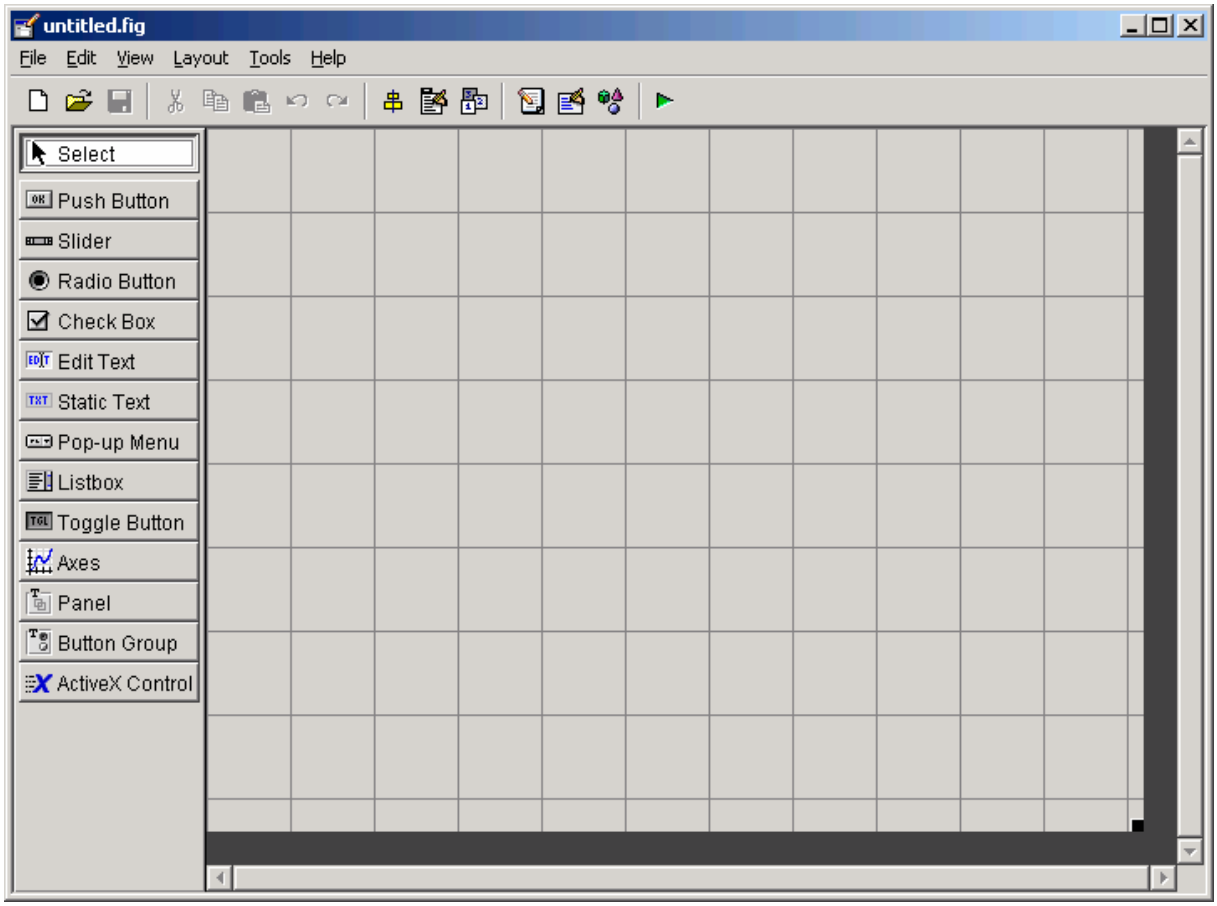


Abbildung 6.1: GUIDE Layout Editor

GUIDE generiert nach Abspeichern der erstellten GUI zwei Dateien mit dem eingegebenen Namen, eine mit der Endung **.fig*, welche die Grafik mit der Anordnung der erstellten Elemente enthält und die andere mit der Endung **.m*, welche die so genannten *Callbacks* dieser Elemente enthält. Jedem Element werden *Callbacks* zugeordnet. Diese Funktionen werden dann beispielsweise beim Anklicken des Buttons oder beim Ändern des Textes ausgeführt und können wiederum sämtliche MATLAB-Befehle enthalten. Außerdem gibt es noch einige grundsätzliche *Callback*-Funktionen, unter anderem die *OpeningFcn* und die *CloseRequestFcn*, die beim Öffnen bzw. Schließen des Programms ausgeführt werden. Somit enthält das generierte *m-File* eine Sammlung von Funktionen, welche jeweils durch ein bestimmtes Ereignis ausgelöst werden. Die Elemente einer GUI lassen sich mit dem *Property-Inspector* (siehe Abbildung 6.2) konfigurieren und den persönlichen Anforderungen anpassen. So können hier von Name über Farbe bis zu Vererbungsinformationen sämtliche Einstellungen gemacht werden.

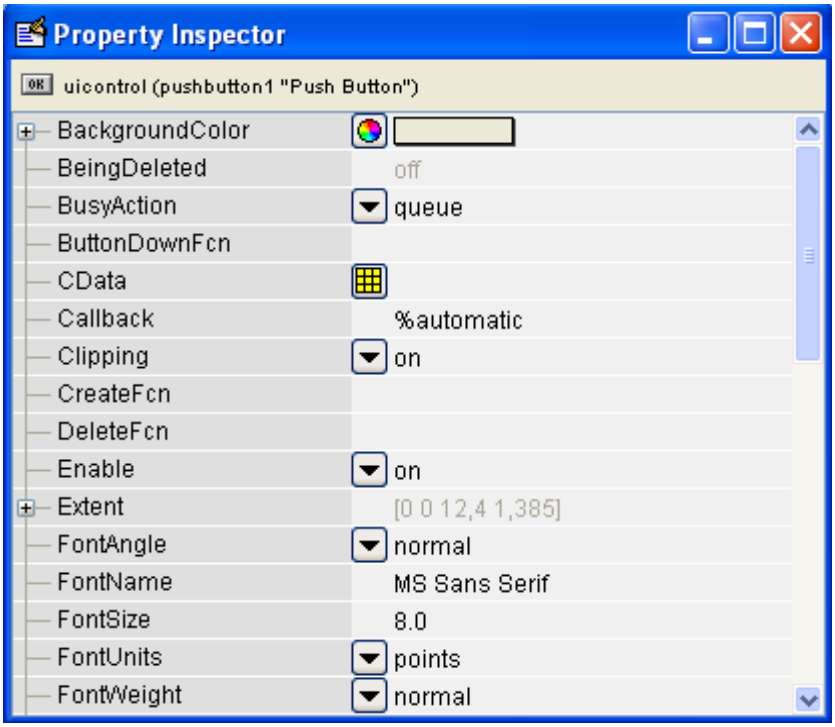


Abbildung 6.2: Der GUIDE-Property-Inspector

Ein mit GUIDE erstelltes GUI lässt sich durch Eingabe des vergebenen Namens im *Command-Window* sowie im *m-File* starten oder durch Drücken des *Play*-Tasters im Layout-Editor testen.

6.1.2.2 SIMULINK und *s-Funktion*

Eine spezielle Toolbox von MATLAB ist SIMULINK, die aus einer Sammlung vorgefertigter *m-Files* besteht. SIMULINK ist die spezielle Toolbox für Modellierung, Simulation und Analyse dynamischer Systeme. Sie unterstützt lineare und nicht-lineare Systeme, die im kontinuierlichen, diskreten Bereich oder einen hybriden Bereich modelliert sind. Mit SIMULINK ist eine graphische Zusammenstellung von Modellen aus vorgefertigten oder selbst zu definierenden Blöcken möglich, die dann simuliert werden können. SIMULINK ist in der Grundversion von MATLAB nicht enthalten, sondern muss hinzugekauft werden. Der Aufruf von SIMULINK erfolgt durch Anklicken des Tasters *simulink* von der *Toolbar* aus oder im MATLAB *Command-Window* durch Eingabe von

>>simulink

Danach öffnet sich eine graphische Oberfläche, der *Simulink Library Browser*. Er stellt ein eigenes Windows-Fenster dar und enthält eine Menüliste und Icons für die Blockbibliotheken. Die am häufigsten verwendete Blöcke sind **Sources** (Quellen), **Sinks** (Senken), **Discrete** (Zeitdiskret), **Continuous** (Zeitkontinuierlich), **Math Operations** (Mathematische Operationen) und **Signal Routing** (Signale Routine). Diese Blockbibliotheken können durch Mausklicken in das Simulationsfenster gezogen werden. Eine genauere Beschreibung der einzelnen Blockbibliotheken ist im SIMULINK-Handbuch und in der Onlinehilfe zu finden.

Eine *s-Funktion* ist eine Beschreibung eines Simulinkblocks in einer Computersprache. Sie kann in MATLAB, C, C++, Ada oder FORTRAN geschrieben werden. *s-Funktionen*, die in C, C++, Ada und FORTRAN geschrieben sind, müssen mit dem MATLAB Dienstprogramm *mex* in MEX-Datei kompiliert werden und beim Aufruf dynamisch in MATLAB gelinkt werden.

Die *s-Funktion* erlaubt den Benutzern eigene Blöcke zu definieren und in SIMULINK Modelle mit vorgefertigten Systemblöcken zusammen zu simulieren. Der Benutzer kann seine eigenen Blöcke nach einfachen Regeln bilden und den Algorithmus in die *s-Funktion* implementieren. Danach kann die *s-Funktion* in einen *s-Funktion*-Block (unter Blockbibliothek User-Defined-Functions) angeordnet und anschließend simuliert werden. Der Benutzer kann auch die Parameter durch ein Parameterfenster in die *s-Funktion* eingeben und die Schnittstelle durch *Masking* nach eigenen Wünschen anpassen. Für eine genauere Beschreibung von SIMULINK und *s-Funktion* wird auf die sehr ausführliche Onlinehilfe verwiesen.

6.2 Programmstrukturen

Hier werden die einzelnen Programmstrukturen der entwickelten Softwaremodule vorgestellt. Dabei unterteilt sich sämtliche Software in zwei Bereiche: den Bereich des Mikroprozessors in C und den Bereich der Regler in MATLAB/SIMULINK. Zu Beginn wird die Programmstruktur des Mikroprozessors erklärt.

6.2.1 Programmstrukturen des Mikroprozessors

In dieser Arbeit, werden die folgenden Hardwarekomponenten des ATmega128ls verwendet:

- I/O Port
    1. PortA Pin (0-6): C2RB (0-6) Counter to Register Bus (7 bit)

    2. PortB Pin (4-7): CPB (0-3) Control Panel Bus (4 bit)

    3. PortC Pin (0-4): CSB (0-4) Counter Select Bus (5 bit)
       Pin 7: CRST Counter Reset Signal

    4. PortE Pin 4: Eingang des umgewandelten Trittfrequenzsignals
       Pin 5: Eingang des umgewandelten Herzfrequenzsignals
       Pin 6: Ausgang des PWM-Signals
       Pin 7: Eingang der externen Hardware-Unterbrechung 7

    5. PortF Pin (0-7): DB Daten Bus (7 bit)

    6. PortG Pin (0-2): CPB (4-6) Control Panel Bus (3 bit)

- Timer / Counter
    1. Timer1: 16-Bit Counter für Tastverhältnis des PWM-Signals

    2. Timer3: 16-Bit Counter für Abtastzeit von 0,25 Sekunden

- Unterbrechungen
    1. Timer1 Überlauf-Unterbrechung
    2. Timer3 Überlauf-Unterbrechung
    3. Externer Hardware-Unterbrechung 7
    4. USART0 RXC-Unterbrechung

Bevor die Bausteine des Mikroprozessors eingesetzt werden können, soll eine Initialisierung für alle Bausteine sowie die MCU (MicroController Unit) durchgeführt werden. Dafür wird zunächst die Struktur des Initialisierungsprogramms vorgestellt.

6.2.1.1 Initialisierung

Die Initialisierungsfolge für alle Bausteine ist frei. Dennoch ist es unbedingt notwendig, dass bevor der Initialisierungsvorgang beginnt, alle Unterbrechungen gesperrt werden. In dieser Arbeit wird der Initialisierungsvorgang in folgender Reihenfolge durchgeführt: Unterbrechungsgesperre, Initialisierung der I/O-Ports, des Timer1, des Timer3, des USART0 und der MCU, Zurücksetzen aller Zähler, Register und Variablen. Der detaillierte Initialisierungsablauf ist in der Abbildung 6.3 zu sehen.

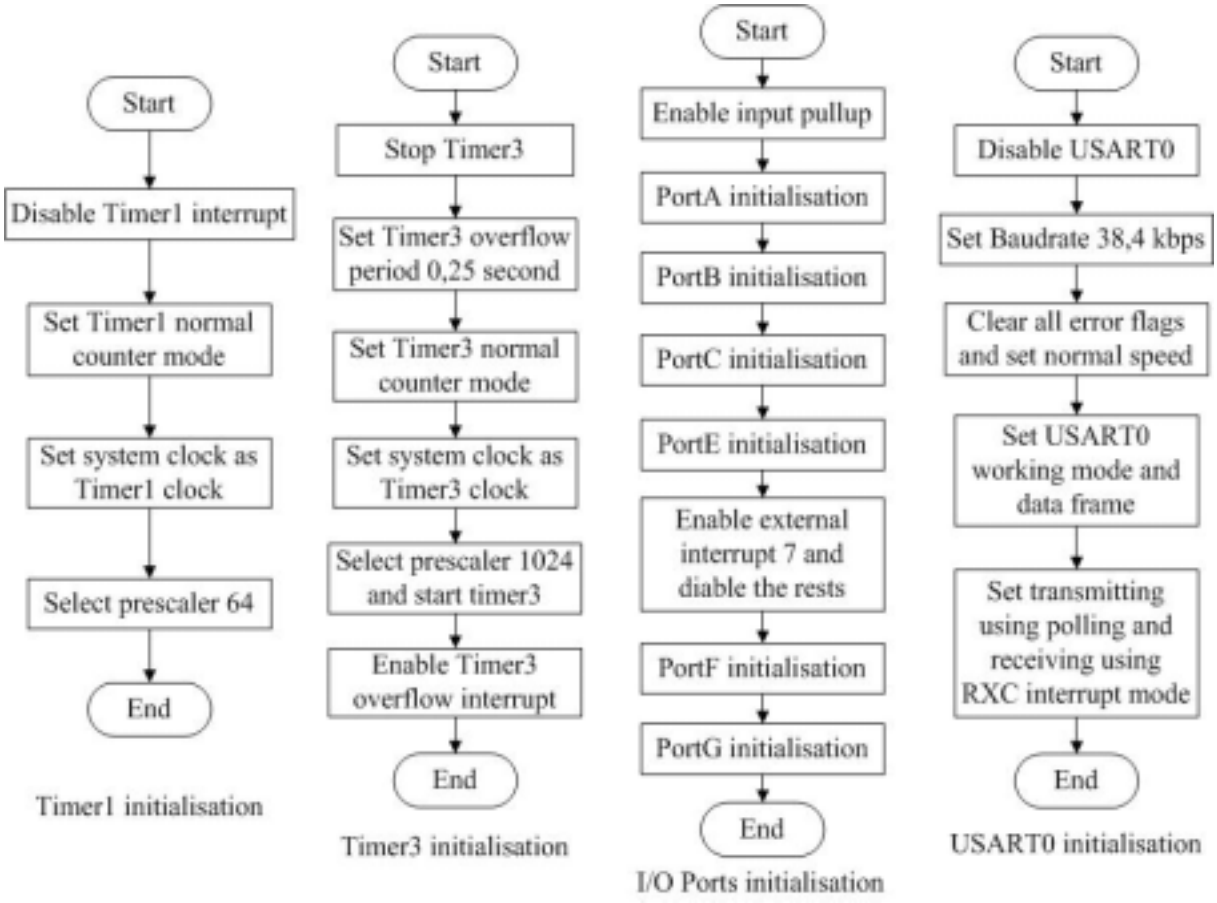


Abbildung 6.3: Initialisierungsablauf aller Bausteine des ATmega128L

Die I/O-Ports werden nach den Pin-Belegungen der Sensorplatine und den Anforderungen der Port-Funktionen initialisiert. Es ist wichtig, dass die Pull-Up Eigenschaften aller Pins aktiviert werden. Ansonsten können keine Signale vom Trittfrequenz- und Geschwindigkeitssensoren eingelesen werden. Der Timer1 und der Timer3 arbeiten im normalen 16-Bit Counter-Mode, ihre Überlauf-Unterbrechungen sind aktiviert. Als Timer-Clock wird der Systemquarz von 7,3728 MHz für beide Timer verwendet. Der einzige Unterschied zwischen beiden Timern ist, dass der Timer1 einen Vorteiler von 64 und der Timer3 von 1024 hat.

Für den seriellen Datenaustausch zwischen Mikroprozessor und PC, wird USART0 mit einer Baudrate von 38,4 kbps initialisiert. Er arbeitet im asynchronen Mode mit einem Datenframe, das aus einem Start Bit, acht Datenbits und einem Stop Bit aufgebaut ist. Es werden keine Parität und keine handshake Bits für die Datenübertragung verwendet. Um korrekte Sensordaten zu erfassen, werden alle Zähler und beteiligte Register vor dem Messstart zurückgesetzt.

Nach der Initialisierung aller Bausteine, kann das Global-Interrupt-Enable-Flag gesetzt werden, um die initialisierten Unterbrechungen zu aktivieren. Dann kann die Main-Schleife des Hauptprogramms ausgeführt werden.

6.2.1.2 Main-Schleife des Hauptprogramms

Das Hauptprogramm des Mikroprozessors hat folgende drei Aufgaben:

1. Sensordatenerfassung und Auswertung

2. Datenaustausch mit dem PC über die serielle Schnittstelle

3. Umwandlung des von PC empfangenen Indexwerts der PWM-Tabelle in das PWM-Signal zur Steuerung der Wirbelstrombremse

Die drei Aufgaben werden in der Main-Schleife des Hauptprogramms realisiert. Dafür müssen in der Main-Schleife alle Funktionsteile anhand der Hardwarekomponenten koordiniert werden, z.B. Zeitplanung für die Datenerfassung von verschiedenen Sensoren, Synchronisation des PWM-Signals mit dem 50-Hz-Sinus-Signal von der Wirbelstrombremse. Um alle Funktionsteile gut zu koordinieren, wird die Main-Schleife hier flagbasiert realisiert. Das heißt, jedem Funktionsteil wird ein Flag zugeordnet. Wenn ein Funktionsteil als Nächstes abzuarbeiten ist, wird das für diesen Teil zugeordnete Flag auf ’1’ gesetzt. Die Main-Schleife fragt alle Flags ständig in einer festen Reihenfolge ab und arbeitet die Teile mit Flag-Wert von ’1’ hintereinander ab. Nach Abarbeitung des Funktionsteils, wird sein Flag auf ’0’ zurückgesetzt. Die Funktionsteile, die höhere Priorität haben oder mit einer bestimmten Periode abgearbeitet werden müssen, werden als Unterbrechungen realisiert. Dafür werden hier vier Unterbrechungen verwendet. Dies sind die externe Hardware-Unterbrechung 7, die USART0 RXC-Unterbrechung, die Timer1- sowie die Timer3-Überlauf-Unterbrechungen. Die Programmabläufe der ISRs dieser Unterbrechungen werden später noch erklärt.

In der Abbildung 6.4 ist das Ablaufdiagramm der Main-Schleife dargestellt. Nach der Initialisierung aller verwendeten Hardwarekomponenten, wird das Global-Interrupt-Enable-Flag auf ’1’ gesetzt, um die initialisierten Unterbrechungen zu aktivieren. Danach beginnt die Main-Schleife. In der Main-Schleife, wird als Erstes abgefragt, ob der Empfangspuffer leer ist.

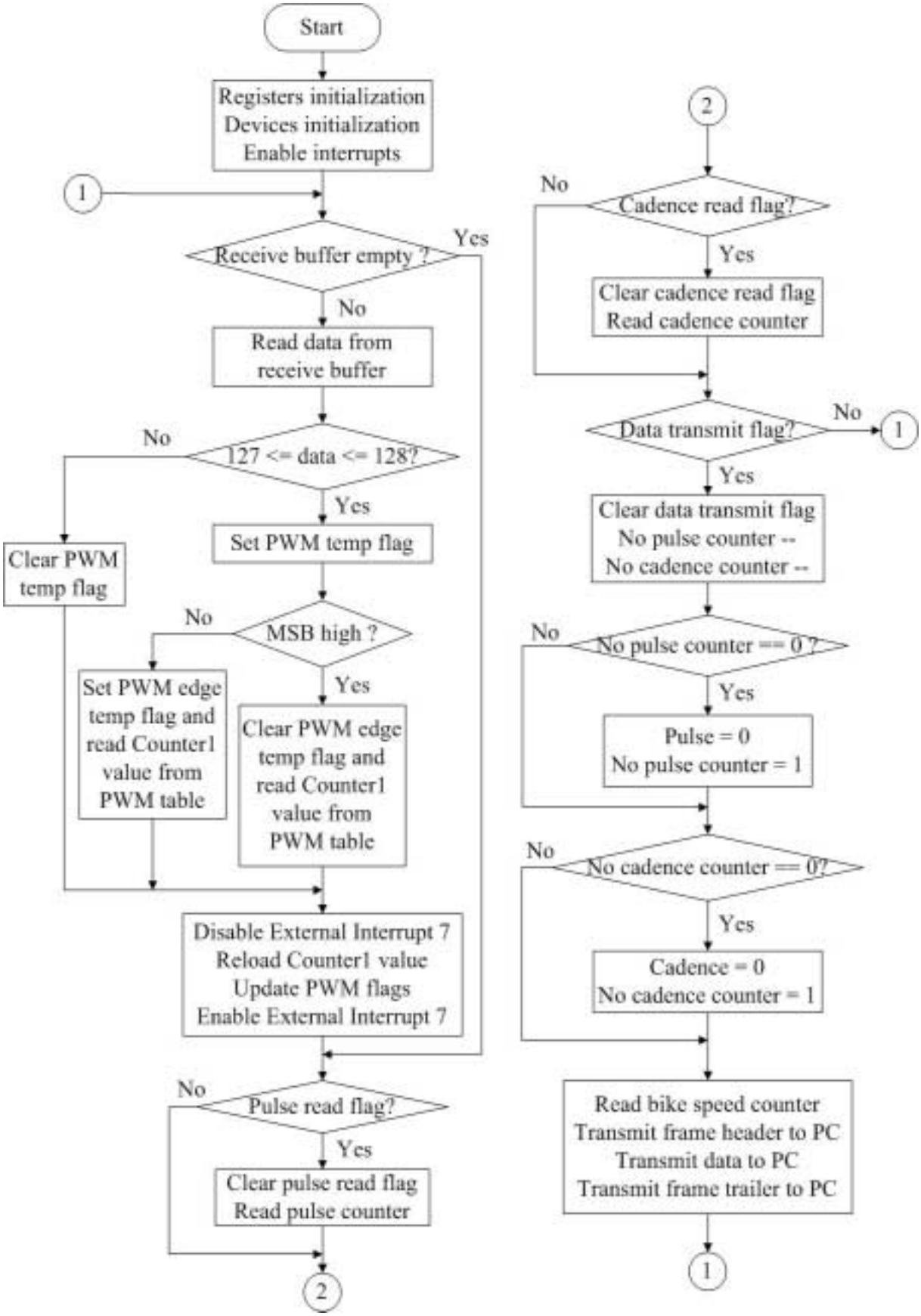


Abbildung 6.4: Programmablauf der Main-Schleife

Ist dies nicht der Fall, so läuft das Programm zur Bearbeitung dieses empfangenen Datums. Zuerst wird das Datum vom Empfangspuffer ausgelesen und dann abgefragt, ob es zwischen 127 und 128 liegt. Wenn dies der Fall ist, bedeutet dies, dass kein PWM-Signal auszugeben ist. Dann wird ein temporäres PWM-Flag auf ’0’ zurückgesetzt. Wenn das Datum nicht zwischen 127 und 128 liegt, bedeutet dies, dass ein PWM-Signal auszugeben ist. Dann wird das temporäre PWM-Flag auf ’1’ gesetzt und weiter abgefragt, ob das Datum größer als 128 ist. Ist dies der Fall, so wird die Pulsbreite des PWM-Signals aus der PWM-Tabelle ausgelesen und ein temporäres PWM-Flanke-Flag auf ’0’ gesetzt, um das PWM-Signal später in ISR der externen Hardware-Unterbrechung 7 bei der negativen Flanke des 50-Hz-Sinus-Signals auszugeben. Wenn das Datum nicht größer als 128 ist, dann ist es kleiner als 127 und das PWM-Signal wird später in ISR der externen Hardware-Unterbrechung 7 bei der positiven Flanke des 50-Hz-Sinus-Signals ausgegeben. Die Pulsbreite des PWM-Signals wird ebenfalls aus der PWM-Tabelle ausgelesen und das temporäre PWM-Flanke-Flag auf ’1’ gesetzt. Nach der Bearbeitung des empfangenen Datums wird die externe Hardware-Unterbrechung 7 deaktiviert, um alle Flags des PWM-Signals und Variablen der Pulsbreite des PWM-Signals zu aktualisieren. Danach wird die externe Hardware-Unterbrechung 7 wieder aktiviert, um das eingestellte PWM-Signal auszugeben.

Wenn der Empfangspuffer leer ist, wird weiter abgefragt, ob das Sensordatum des Herzfrequenzzählers bereit zum Auszulesen ist. Ist dies der Fall, so wird das Sensordatum vom Herzfrequenzzähler ausgelesen und für die Übertragung zum PC in eine bestimmte Variable gespeichert. Wenn kein Herzfrequenzdatum auszulesen ist, wird weiter abgefragt, ob das Sensordatum des Trittfrequenzzählers bereit zum Auszulesen ist. Ist dies der Fall, so wird das Sensordatum vom Trittfrequenzzähler ausgelesen und ebenfalls für die Übertragung zum PC in eine bestimmte Variable gespeichert. Wenn kein Trittfrequenzdatum auszulesen ist, wird weiter abgefragt, ob das Datenübertragungsflag ’1’ ist. Ist dies nicht der Fall, so läuft die Schleife zurück zur Abfragen des Empfangspuffers.

Ist das Datenübertragungsflag ’1’, so wird zuerst das Datenübertragungsflag auf ’0’ zurückgesetzt. Dann werden die Zähler für nicht vorhandene Herzfrequenz und nicht vorhandene Trittfrequenz bearbeitet. Die beiden Zähler dienen jeweils zur Erkennung der Situation, wenn kein Trittfrequenz- oder Herzfrequenzsignal von den Sensoren kommt. Zuerst werden beide Zähler um 1 erniedrigt. Danach wird abgefragt, ob der Zähler für nicht vorhandene Herzfrequenz null ist. Ist dies der Fall, so weiß man, dass kein Herzfrequenzsignal vom Sensor kommt. Damit wird die Herzfrequenzvariable auf ’0’ zurückgesetzt und der Zähler für die Subtraktion der nächsten Schleife auf ’1’ gesetzt. Wenn der Zählerzustand ungleich null ist, wird weiter abgefragt, ob der Zähler für nicht vorhandene Trittfrequenz null ist. Ist dies der Fall, so ist die Bearbeitung der beiden Zähler abgeschlossen. Wenn der Zähler null ist, so weiß man, dass kein Trittfrequenzsignal vom Sensor kommt. Damit wird ebenfalls die Trittfrequenzvariable auf ’0’ zurückgesetzt und der Zähler für die Subtraktion der nächsten Schleife auf ’1’ gesetzt. Danach ist die Bearbeitung der beiden Zähler beendet.

Im Anschluss daran wird das Sensordatum der Radgeschwindigkeit vom Geschwindigkeitszähler ausgelesen und daraufhin alle Sensordaten im Polling-Mode mit dem entwickelten Kommunikationsprotokoll an den PC gesendet. Nach dem Datenversand zum PC ist die Main-Schleife einmal vollständig durchgeführt. Danach beginnt die Schleife wieder mit der Abfrage des Empfangspuffers.

Der einzelne Programmablauf für die Sensordatenerfassung, den Datenversand zum PC und den Datenempfang vom PC wird in den folgenden Abschnitten ausführlich vorgestellt.

6.2.1.3 Auslesen des Zählers der Sensordaten

Wie zuvor erwähnt, sind drei Arten von Sensordaten zu erfassen, die Tritt- und Herzfrequenz des Radfahrers und die Geschwindigkeit des Rades. Zuerst wird das Auslesen des Geschwindigkeitszählers vorgestellt. Die Signal-Impulse des Geschwindigkeitssensors werden in einem 8-Bit-Zähler gezählt. Der Zählerzustand wird alle 0,25 Sekunden periodisch ausgelesen, um die aktuelle Geschwindigkeit zu berechen. Das Ablaufdiagramm zum Auslesen des Geschwindigkeitszählers ist in Abbildung 6.5 dargestellt.

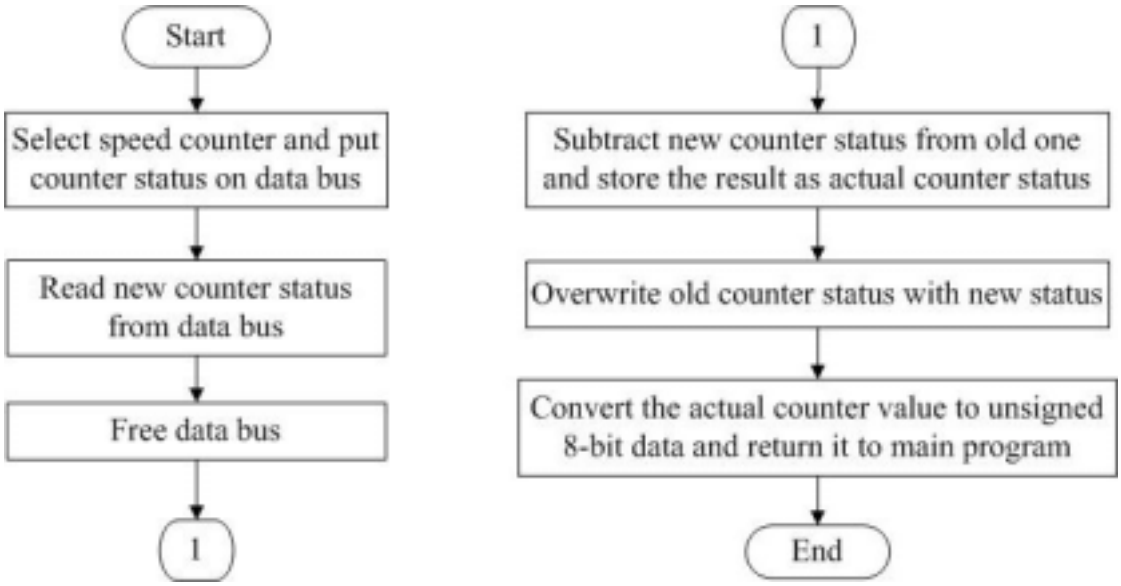


Abbildung 6.5: Ablaufdiagramm zum Auslesen des Geschwindigkeitszählers

Zuerst wird der Geschwindigkeitszähler ausgewählt und das Datum auf den Datenbus gelegt. Danach wird der neue Zählerzustand vom Datenbus eingelesen und der Datenbus freigegeben. Der aktuelle Zählerzustand wird durch Subtraktion des alten Zählerzustandes vom neuen berechnet. Im Anschluss wird der alte Zählerzustand durch den neuen überschrieben. Zum Schluss wird der aktuelle Zählerzustand zu einem 8-Bit-Unsigned-Datum konvertiert und an die Main-Schleife zurückgegeben.

Für die Messung der Herz- und Trittfrequenz ,wie im Konzeptentwurf der Sensordatenerfassung erklärt, werden jeweils drei 8-Bit-Zähler, die zusammen einen 24-Bit-Zähler bilden, verwendet. Nach dem Auslesen des 24-Bit-Zählerzustandes, wird der alte Zählerzustand von dem neuen abgezogen um die aktuelle Herz- bzw. Trittfrequenz zu berechnen. Die berechneten Frequenzwerte werden für die Übertragung zum PC in bestimmten Variablen gespeichert. Als Beispiel wird das Ablaufdiagramm zum Auslesen des Trittfrequenzzählers in Abbildung 6.6 dargestellt. Der Ablauf zum Auslesen des Herzfrequenzzählers ist ähnlich dem des Trittfrequenzzählers mit der einzelnen Ausnahme, dass der Herzfrequenzzähler vor dem Auslesen ausgewählt werden muss.

Zuerst wird der MSB-Zähler (Most Significant Byte) ausgewählt und dessen Datum auf den Datenbus gelegt. Dann wird das Datum vom Datenbus eingelesen und in eine Variable gespeichert. Im Anschluss werden die Daten vom Zähler des mittleren Bytes und vom LSB-Zähler (Least Significant Byte) ausgelesen und mit dem Datum des MSB zusammen zu einem

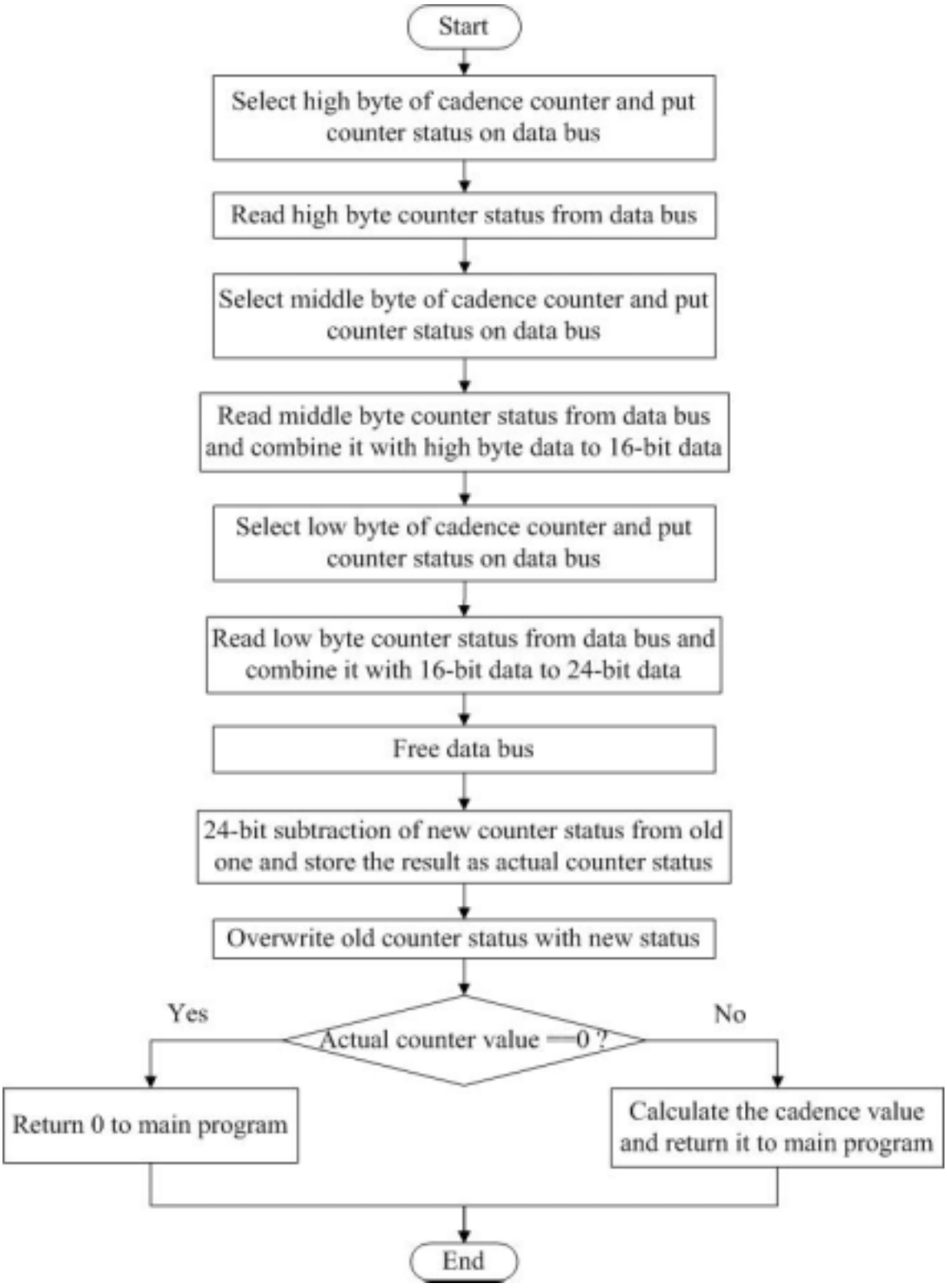


Abbildung 6.6: Ablaufdiagramm zum Auslesen des Trittfrequenzzählers

24-Bit-Datum kombiniert. Danach wird der Datenbus freigegeben und der aktuelle Zählerzustand durch eine 24-Bit-Subtraktion des alten Zählerzustandes vom neuen berechnet. Schließlich wird der alte Zählerzustand durch den neuen überschrieben. Daraufhin wird abgefragt, ob der aktuelle Zählerzustand null ist. Ist dies nicht der Fall, so wird die Trittfrequenz

durch Division des Systemquarzes durch den aktuellen Zählerzustand ermittelt und anschließend an die Main-Schleife zurückgegeben. Wenn der aktuelle Zählerzustand null ist, wird eine Null als aktuelle Trittfrequenz an die Main-Schleife zurückgegeben, weil eine Null nicht als einen Divisor dienen kann.

Um den Zählerzustand des Zählerbausteins 74HC590 auszulesen, muss der Zählerzustand vor dem Auslesen in das Register geladen werden. Das Laden des Zählerzustandes in das Register wird in den ISRs der Unterbrechungen erledigt.

6.2.1.4 ISRs der Unterbrechungen

Wie zuvor erwähnt, werden in dieser Arbeit vier Unterbrechungen verwendet. Dies sind die externe Hardware-Unterbrechung 7, die USART0 RXC-Unterbrechung, die Timer1- sowie die Timer3-Überlauf-Unterbrechungen. Es werden im Folgenden die im Hintergrund laufenden ISRs der vier Unterbrechungen vorgestellt. Als Erstes wird die ISR der externen Hardware-Unterbrechung 7 betrachtet.

Die externe Hardware-Unterbrechung 7 wird an PortE-Pin 7 ausgelöst. Die Hauptaufgabe dieser Unterbrechung ist die Steuerung der Wirbelstrombremse mit einem PWM-Signal. Um die Wirbelstrombremse zu steuern, muss das PWM-Signal mit dem aus der Wirbelstrombremse kommenden Sinus-Signal so synchronisiert werden, dass eine positive Flanke des PWM-Signals nur beim Null-Durchgang des Sinus-Signals ausgegeben wird. Dafür wird das 50-Hz-Sinus-Signal in ein Rechtecksignal umgewandelt und das resultierende Rechtecksignal als Unterbrechungsquelle mit PortE-Pin 7 verbunden. Die Umwandlung erfolgt so, dass bei jedem Null-Durchgang des Sinus-Signals, die Flanke des Rechtecksignals einmal wechselt. Um die Ausgabe der positiven Flanken des PWM-Signals an beiden Flanken (positive und negative) des Rechtecksignals zu ermöglichen, wird die Unterbrechung 7 so konfiguriert, dass sie an beiden Flanken auslösbar ist. Damit hat die Unterbrechung 7 eine Frequenz von 100 Hz. Mit dieser Frequenz werden andere Aufgaben in die ISR der Unterbrechung 7 erfüllt. Die von Herz- und Trittfrequenz umgewandelten Signale werden in diese ISR geprüft, ob die entsprechenden Zählerzustände in die Register geladen werden können. Ist dies der Fall, so wird ein Register-Laden-Signal auf dem C2RB-Bus ausgegeben, um die entsprechenden Zählerzustände in die Register zu laden. Hierbei ist es unbedingt notwendig, dass das Register-Laden-Signal nach seiner Ausgabe auf dem C2RB-Bus mindestens einen Befehltakt anstehen muss, um den Zählerzustand in das Register laden zu können.

Das Ablaufdiagramm der ISR der Hardware-Unterbrechung 7 ist in Abbildung 6.7 dargestellt. Wie bereits zuvor erläutert, sind Pin 4, 5, 6, 7 der PortE jeweils der Eingang des umgewandelten Trittfrequenzsignals, der Eingang des umgewandelten Herzfrequenzsignals, der Ausgang des PWM-Signals und der Eingang der externen Hardware-Unterbrechung 7. Dafür wird der Zustand von PortE am Start der ISR eingelesen, um später die Pinpegel zu prüfen. Zuerst wird abgefragt, ob das PWM-Flag gesetzt ist. Ist dies der Fall, so bedeutet dies, dass ein PWM-Signal auszugeben ist. Dann wird der Pegel an PortE-Pin 7 geprüft. Ist der Pegel hoch, so bedeutet dies, dass eine positive Flanke die Unterbrechung 7 ausgelöst hat. Ist der Pegel niedrig, so bedeutet dies, dass eine negative Flanke die Unterbrechung 7 ausgelöst hat. Die Ausgabe der positiven Flanke des PWM-Signals bei der positiven oder negativen Flanke an PortE-Pin 7 entspricht einer unterschiedlichen Kraft der Wirbelstrombremse. Deshalb wird folgendermaßen vorgegangen: um die geforderte Kraft an der Wirbelstrombremse zu erzeugen, wird

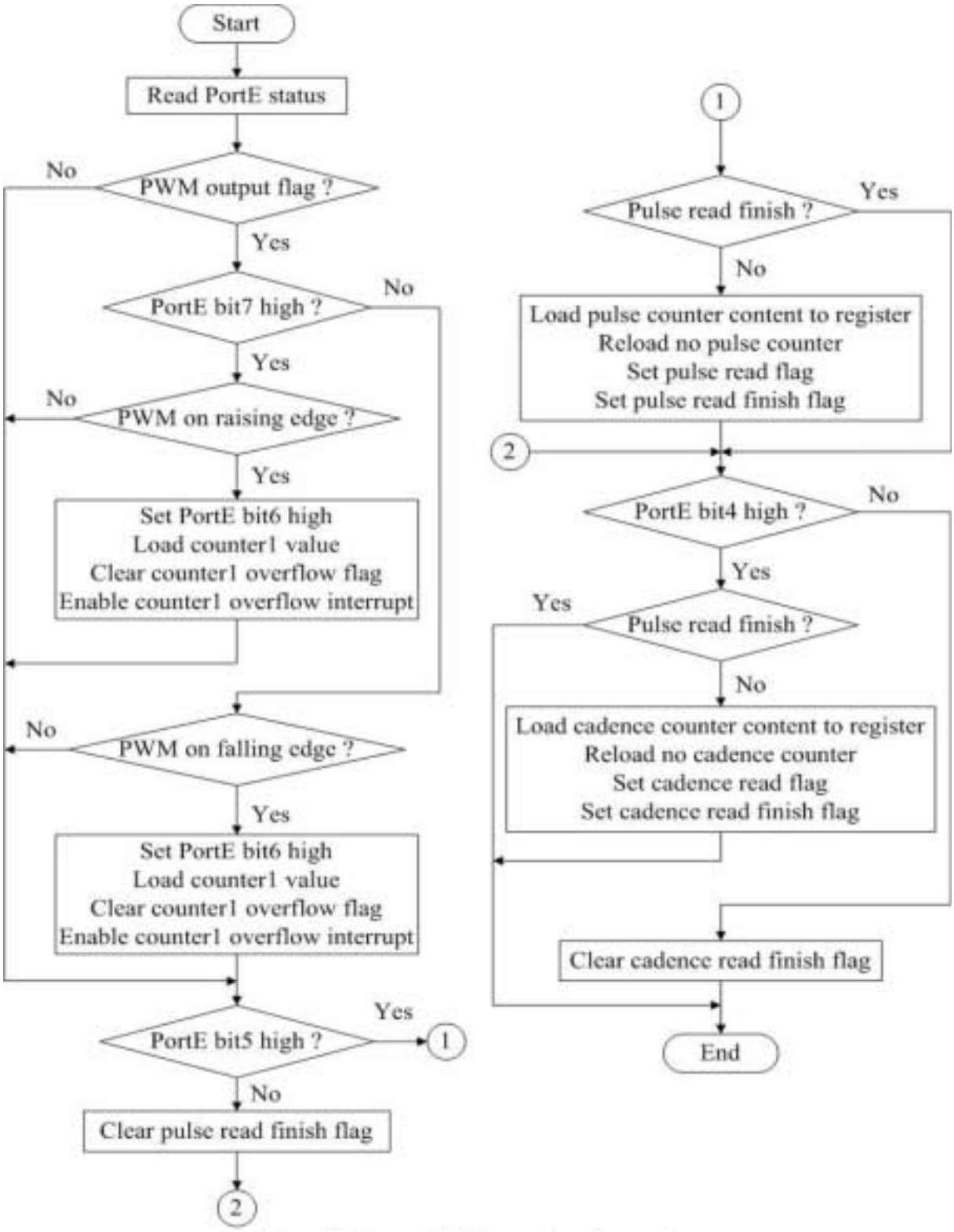


Abbildung 6.7: Ablaufdiagramm der ISR der externen Hardwareunterbrechung 7

bei der positiven oder negativen Flanke an PortE-Pin 7 weiter abgefragt, ob das PWM-Flanke-Flag gesetzt ist. Wenn das Flag gesetzt ist, muss die positive Flanke des PWM-Signals bei der positiven Flanke an PortE-Pin 7 ausgegeben werden. Ist das Flag nicht gesetzt, so

muss die positive Flanke des PWM-Signals bei der negativen Flanke an PortE-Pin 7 ausgegeben werden. Wenn die Flanke an PortE-Pin 7 mit der im PWM-Flanke-Flag präsentierten Flanke übereinstimmt, wird die positive Flanke des PWM-Signals an PortE-Pin 6 ausgegeben und der Timer1, der die PWM-Pulsbreite zählt, mit entsprechender Überlaufsperiode gestartet. Danach wird das Anforderungsflag der Überlauf-Unterbrechung des Timer1 gelöscht und die Überlauf-Unterbrechung wieder aktiviert. Wenn die im Timer1 eingestellte Zeit abgelaufen ist, wird die Überlauf-Unterbrechung des Timer1 ausgelöst, um die negative Flanke des PWM-Signals auszugeben. Wenn die Flanke an PortE-Pin 7 mit der im PWM-Flanke-Flag präsentierten Flanke nicht übereinstimmt, wird bei dieser Flanke an PortE-Pin 7 kein PWM-Signal ausgegeben.

Wenn das PWM-Flag nicht gesetzt ist, so bedeutet dies, dass kein PWM-Signal auszugeben ist. Dann wird der Pegel an PortE-Pin 5 geprüft. Ist der Pegel hoch, so wird weiter abgefragt, ob die Herzfrequenzzähler ausgelesen wurden. Wenn dies der Fall ist, so ist die Bearbeitung des Pegels an PortE-Pin 5 beendet. Wenn die Herzfrequenzzähler noch nicht ausgelesen wurden, so wird ein Register-Laden-Signal auf dem C2RB-Bus ausgegeben, um die Zustände der Herzfrequenzzähler in die Register zu laden. Danach wird ein HF-Lesen-Flag auf ’1’ gesetzt, um in der Main-Schleife die Herzfrequenzzähler auszulesen. Ein HF-Lesen-Fertig-Flag wird auch auf ’1’ gesetzt, um der nächsten Unterbrechung mitzuteilen, dass die Herzfrequenzzähler ausgelesen wurden. Dann ist die Bearbeitung des Pegels an PortE-Pin5 auch beendet.

Wenn der Pegel an PortE-Pin 5 niedrig ist, so wird das HF-Lesen-Fertig-Flag auf ’0’ zurückgesetzt und der Pegel an PortE-Pin 4 geprüft. Ist der Pegel hoch, so wird weiter abgefragt, ob die Trittfrequenzzähler ausgelesen wurden. Wenn dies der Fall ist, so ist die ISR der Hardware-Unterbrechung 7 beendet. Wenn die Trittfrequenzzähler noch nicht ausgelesen wurden, so wird ein Register-Laden-Signal auf dem C2RB-Bus ausgegeben, um die Zustände der Trittfrequenzzähler in die Register zu laden. Danach wird ein TF-Lesen-Flag auf ’1’ gesetzt, um in der Main-Schleife die Trittfrequenzzähler auszulesen. Ein TF-Lesen-Fertig-Flag wird auch auf ’1’ gesetzt, um die nächste Unterbrechung darauf hinzuweisen, dass die Trittfrequenzzähler ausgelesen wurden. Dann ist die ISR der Hardware-Unterbrechung 7 beendet. Wenn der Pegel an PortE-Pin 4 niedrig ist, so wird das TF-Lesen-Fertig-Flag auf ’0’ zurückgesetzt und die ISR der Hardware-Unterbrechung 7 ist beendet.

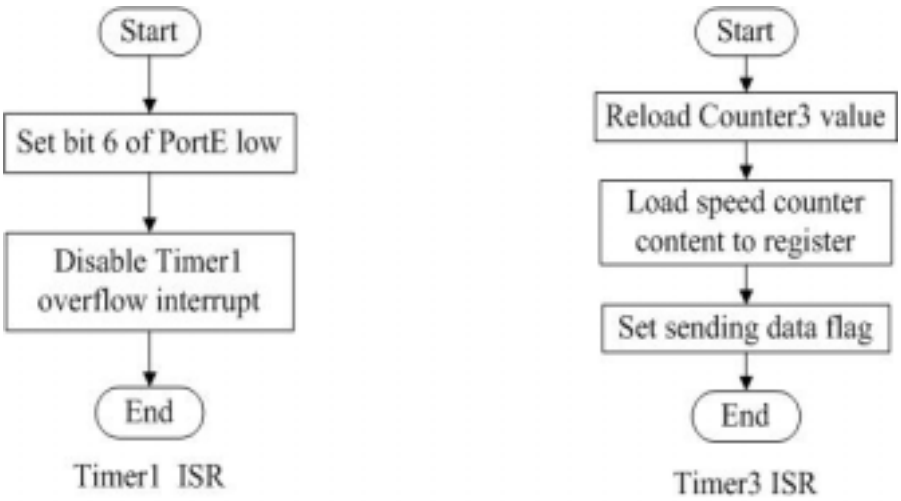
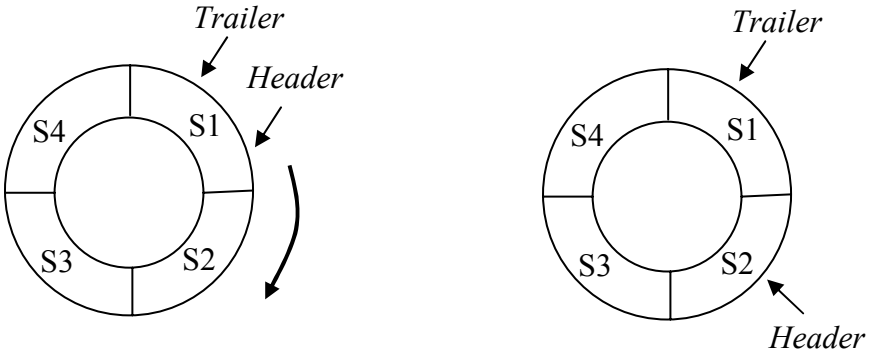


Abbildung 6.8: Ablaufdiagramme der ISRs von Timer1 und Timer3

Als Zweites wird hier die ISR des Timer1 und des Timer3 betrachtet. Die Ablaufdiagramme der beiden ISRs sind in Abbildung 6.8 dargestellt. Der Timer1 wird von der ISR der Hardware-Unterbrechung 7 gestartet und ist für die Steuerung der PWM-Pulsbreite zuständig. Die ISR des Timer1 wird ausgelöst, wenn die eingestellte Zeit für die PWM-Pulsbreite abgelaufen ist. Daraufhin wird die negative Flanke des PWM-Signals in der ISR ausgegeben und die Überlauf-Unterbrechung des Timer1 wieder gesperrt.

Der Timer3 ist für die 0,25 Sekunden Abtastzeit zuständig. Er wird von der Main-Schleife gestartet und seine Überlauf-Unterbrechung ist immer aktiv. Wenn die Überlauf-Unterbrechung des Timer3 ausgelöst wurde, wird die ISR durchgeführt. Zuerst werden die Registerwerte des Timer3 geladen, danach wird der Zustand des Geschwindigkeitszählers in das Register geladen. Zum Schluss wird das Datenübertragungsflag auf ’1’ gesetzt, um der Main-Schleife mitzuteilen, dass die Sensordaten an den PC gesendet werden sollen.

Als Letztes wird die ISR der USART0 RXC-Unterbrechung vorgestellt. Um das Überschreiben der alten Daten durch neue Daten zu verhindern, wird zuerst ein 4-Byte-Puffer mit Ring-Struktur zum Speichern der empfangenen Daten erstellt. Die Struktur des Puffers ist in Abbildung 6.9 dargestellt. Es werden zwei Variablen, *Header* und *Trailer*, verwendet, um auf den Puffer zuzugreifen. Die Variable *Header* dient als Adresszeiger des Puffers zum Schreiben und die Variable *Trailer* zum Lesen. Am Anfang, d.h. bei leerem Speicher, zeigen *Header* und *Trailer* beide auf die erste Speicherzelle *S1* im Puffer. Dies ist in der Abbildung links gezeigt. Wenn ein Datum empfangen wurde, wird *Header* um 1 erhöht und das Datum in die Zelle *S2*, auf die *Header* zeigt, gespeichert. Dies ist in der Abbildung rechts gezeigt. In der Main-Schleife wird daraufhin abgefragt, ob sich im Puffer ein Datum befindet. Ist dies der Fall, so wird *Trailer* um 1 erhöht und das Datum von der Speicherzelle, auf die *Trailer* zeigt, ausgelesen. Wenn kein Datum im Puffer vorliegt, wird ein Puffer-Leer-Flag auf ’1’ gesetzt. Wenn *Header* und *Trailer* durch Erhöhung das Ende des Puffers überschreiten, werden sie wieder auf die erste Speicherzelle des Puffers zurückgesetzt.



leer mit Datum in S2

Abbildung 6.9: Empfangspuffer mit Ring-Struktur

Das Ablaufdiagramm der ISR der RXC-Unterbrechung ist in Abbildung 6.10 links darstellt. Wenn ein Datum empfangen wurde, wird *Header* um 1 erhöht und das Datum in die Zelle, auf die *Header* zeigt, gespeichert. Danach wird das Puffer-Leer-Flag auf ’0’ zurückgesetzt, um darauf hinzuweisen, dass im Empfangspuffer ein Datum vorliegt.

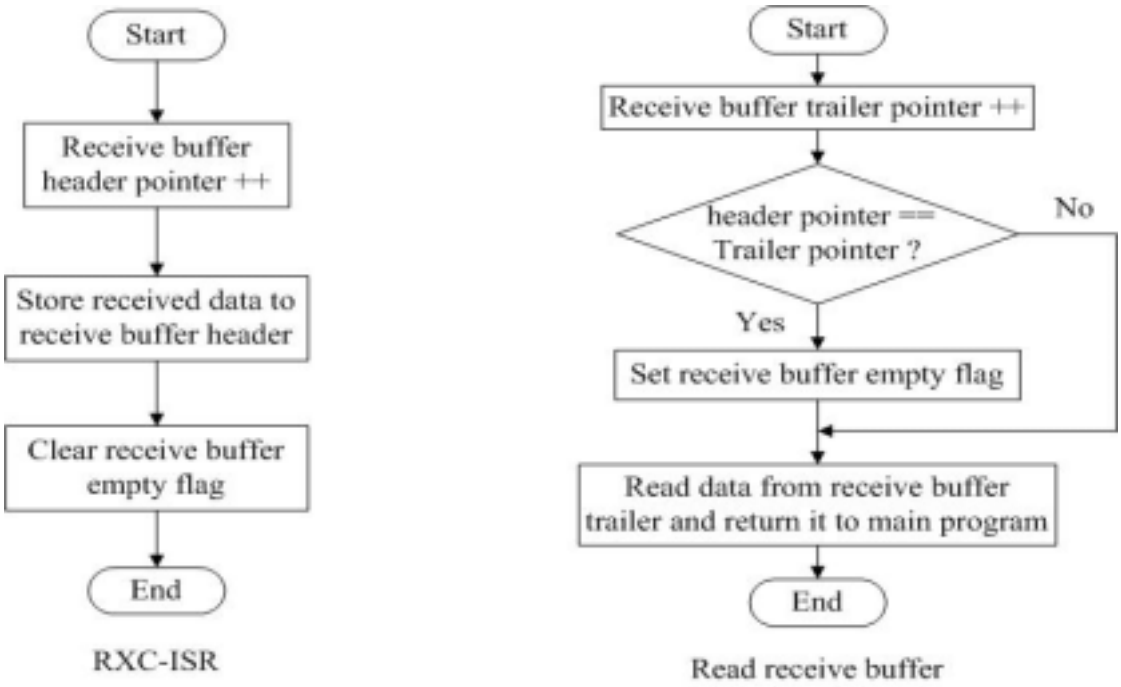


Abbildung 6.10: Ablaufdiagramm des Programms zum Datenempfangen

6.2.1.5 Datenübertragung über USART0

Für die Datenübertragung zwischen Mikroprozessor und PC wird hier USART0, das Hardwaremodul des ATmega128L, verwendet, da USART0 den seriellen Datenfluss steuern kann. In dieser Arbeit wird die RXC-Unterbrechung für den Datenempfang vom PC verwendet, damit die zu empfangenden Daten nicht verloren gehen können. In der Main-Schleife wird abgefragt, ob der Empfangspuffer leer ist (siehe Abbildung 6.4). Wenn dies nicht der Fall ist, wird ein Unterprogramm von der Main-Schleife aufgerufen, um ein Datum aus dem Empfangspuffer zu lesen. Das Ablaufdiagramm des Unterprogramms zum Lesen des Empfangspuffers ist in Abbildung 6.10 rechts dargestellt. Zuerst wird *Trailer* des Empfangspuffers um 1 erhöht und dann wird abgefragt, ob *Trailer* gleich *Header* ist. Ist dies der Fall, so wird das Puffer-Leer-Flag auf ’1’ gesetzt. Wenn *Trailer* ungleich *Header* ist, wird ein Datum von der Speicherzelle, auf die *Trailer* zeigt, ausgelesen und an die Main-Schleife zurückgegeben. So ist das Unterprogramm beendet.

Für den Datenversand zum PC wird hier die Polling-Mode verwendet. Es werden die Tritt- und Herzfrequenz des Menschen sowie die Geschwindigkeit des Rades zum PC gesendet. Um die Sensordaten so schnell wie möglich an den PC zu senden, wird ein 8-Byte-Sendepuffer mit Ring-Struktur erstellt, um die Sensordaten vor dem Versand zum PC zu speichern. Die Struktur des Sendepuffers ist ähnlich der des Empfangspuffers und wird hier nicht mehr erklärt. Genauso wie beim Empfangspuffer, werden hier auch zwei Variablen, *Header* und *Trailer*, verwendet, um auf den Sendepuffer zuzugreifen.

Das Ablaufdiagramm des Unterprogramms zum Speichern der Sensordaten im Sendepuffer ist in Abbildung 6.11 rechts dargestellt. Wenn das Unterprogramm von der Main-Schleife aufgerufen wurde, wird *Header* um 1 erhöht und das zu speichernde Datum in die Zelle, auf

die *Header* zeigt, gespeichert. Dann wird abgefragt, ob *Trailer* gleich *Header* ist. Wenn dies nicht der Fall ist, kehrt das Unterprogramm direkt zur Main-Schleife zurück. Wenn *Trailer* gleich *Header* ist, wird ein Puffer-Voll-Flag auf ’1’ gesetzt und das Unterprogramm kehrt auch zur Main-Schleife zurück.

Der Datenversand zum PC erfolgt mit einer Abtastzeit von 0,25 Sekunden. In der Main-Schleife wird abgefragt, ob das Datenübertragungsflag ’1’ ist (siehe Abbildung 6.4). Ist dies der Fall, so wird ein Unterprogramm von der Main-Schleife aus aufgerufen, um ein Datum von dem Sendepuffer an den PC zu senden. Das Ablaufdiagramm des Unterprogramms zum Datenversand ist in Abbildung 6.11 links dargestellt. Zuerst wird abgefragt, ob der Sendepuffer leer ist. Wenn dies der Fall ist, kehrt das Unterprogramm sofort zur Main-Schleife zurück. Wenn sich Daten im Sendepuffer befinden, wird zuerst das Puffer-Voll-Flag auf ’0’ zurückgesetzt und dann *Trailer* um 1 erhöht. Danach wartet das Programm solange, bis der Sendepuffer von USART0 leer ist. Daraufhin wird ein Datum von der Speicherzelle, auf die *Trailer* zeigt, ausgelesen und in den Sendepuffer von USART0 geschrieben. Damit ist das Unterprogramm zu Ende und kehrt zur Main-Schleife zurück. Das Datum wird automatisch vom Sendepuffer von USART0 am PC gesendet.

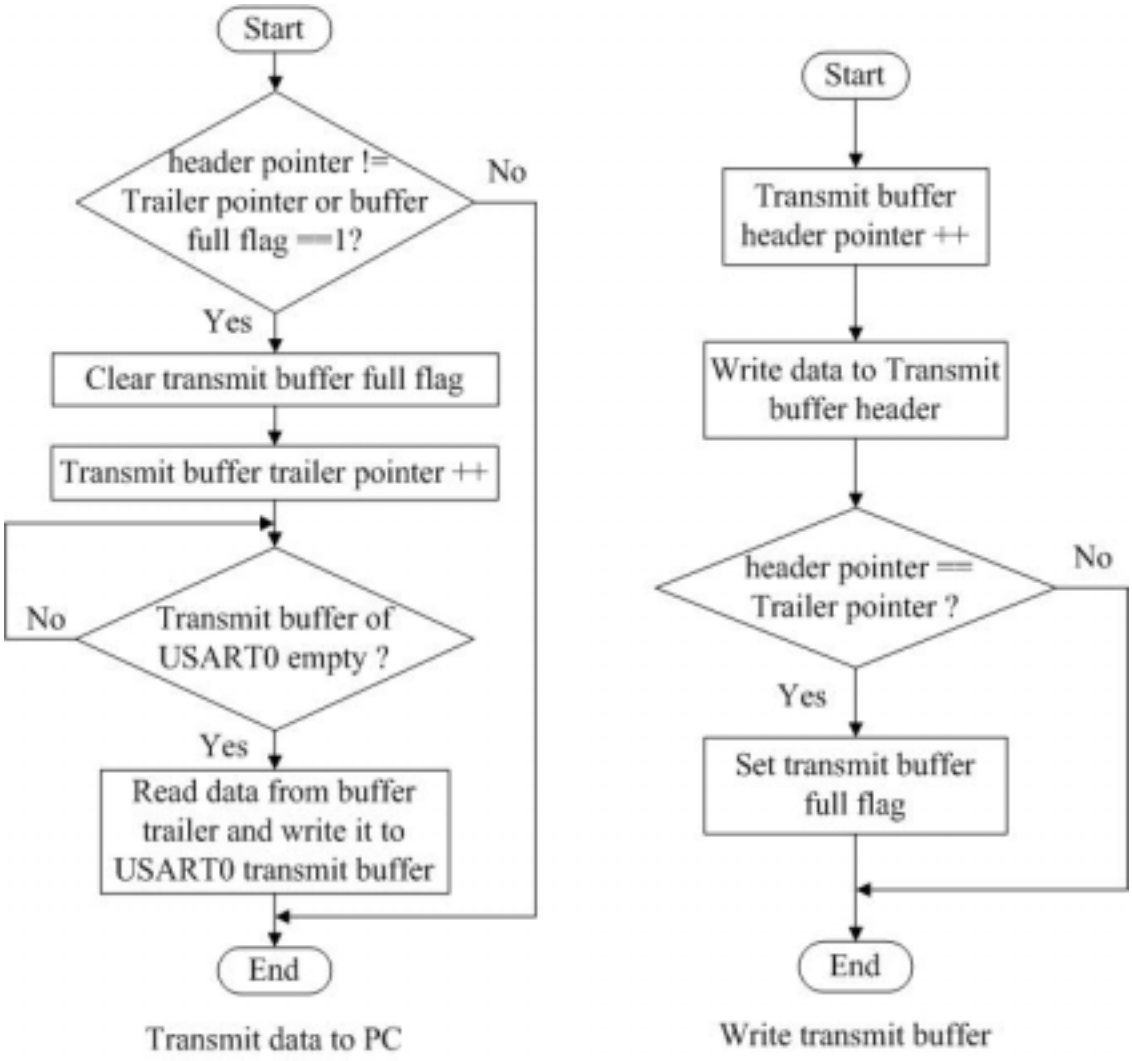


Abbildung 6.11: Ablaufdiagramm des Programms zum Datenversand

6.2.2 Struktur der Programme in MATLAB/SIMULINK

Die Programme, die in MATLAB/SIMULINK implementiert werden, haben die Aufgaben der Datenaustausch zwischen Mikroprozessor und PC durchzuführen und daraufhin die Trainingszustandmeldungen auf dem Display darzustellen. Weiterhin ist die Leistungssteuerung zu implementieren und eine Benutzerschnittstelle zu erstellen. Die Leistungssteuerung wird durch einen zwei-dimensionalen Lookup-Table realisiert, der das Kennfeld der Wirbelstrombremse beschreibt. Um die Leistungssteuerung zu realisieren, muss zuerst die Wirbelstrombremse identifiziert werden, um den Lookup-Table erstellen zu können.

6.2.2.1 Identifikation der Wirbelstrombremse

Die Identifikation der Wirbelstrombremse ist mit dem in Abbildung 4.8 dargestellten Messaufbau durchgeführt worden. Die Bremse wird über eine Koppelung mit einer Messwelle gekoppelt. Das maximale Drehmoment, das mit der verwendeten Welle gemessen werden kann, beträgt 5 Nm. Zur Einstellung der Bremskraft wird der Ergocomputer der Firma Tacx angeschlossen. Die Pulsbreiten, die das Drehmoment der Bremse steuern, werden mit einem Oszilloskop gemessen. Damit ist der direkte Vergleich der vom Ergocomputer abgelesen Werte und der gemessenen Werte möglich. Die Drehzahlen werden so gewählt, dass sich Geschwindigkeiten von 2,5, 5, 10, bis 55 km/h ergeben.

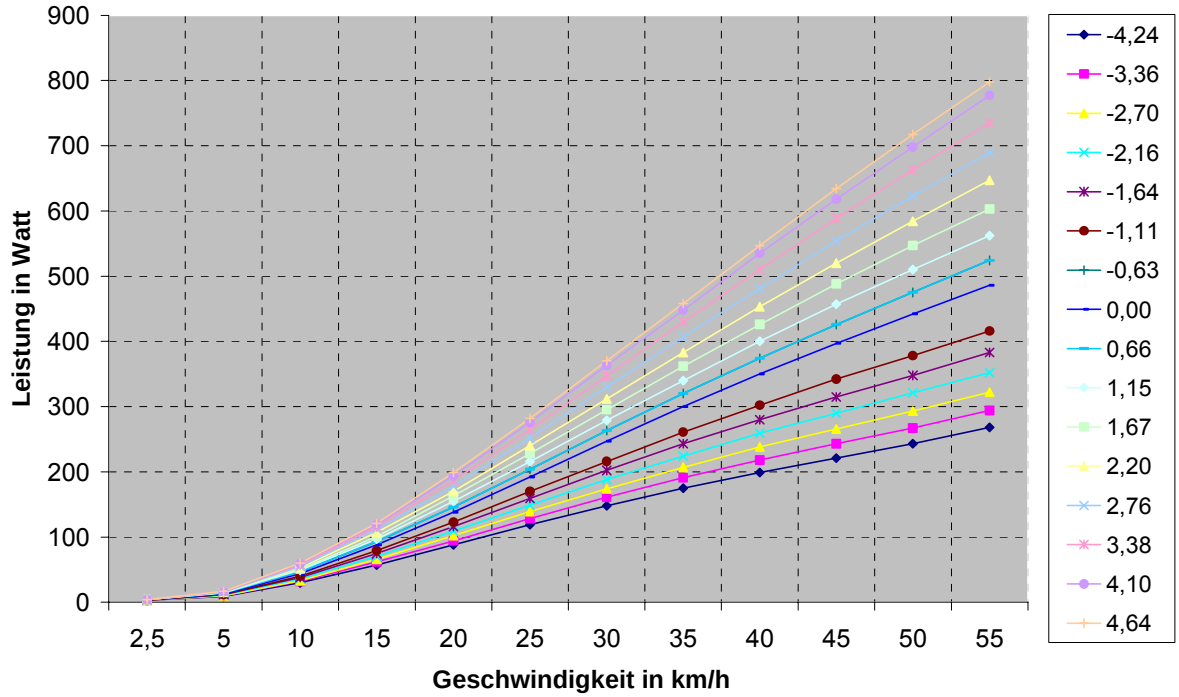


Abbildung 6.12: Kennfeld der Wirbelstrombremse (von Ergocomputer abgelesen)

Die Ergebnisse, die vom Ergocomputer abgelesenen und gemessen wurden, sind in Abbildung 6.12 und Abbildung 6.13 dargestellt. Auf der rechten Seite der Abbildungen stehen die Werte der PWM-Pulsbreite in ms. Das PWM-Signal wird immer beim Null-Durchgang des Sinus-Signals ausgegeben. Hierbei das Minuszeichen bedeutet, dass die Phase des Sinus-Signals ein geradzahliges Vielfaches von π ist. Auf der x und y Achse sind jeweils

die Geschwindigkeit in km/h und die Leistung in Watt aufgetragen. Aus den Abbildungen ist zu erkennen, dass die Leistungen der Wirbelstrombremse für bestimmte PWM-Pulsbreiten mit zunehmender Geschwindigkeit ansteigen. Die Leistungskurven für bestimmte PWM-Pulsbreiten mit zunehmender Geschwindigkeit sind näherungsweise linear oder zumindest stückweise linear. Dennoch sind die Abweichungen zwischen beiden Ergebnissen sehr groß.

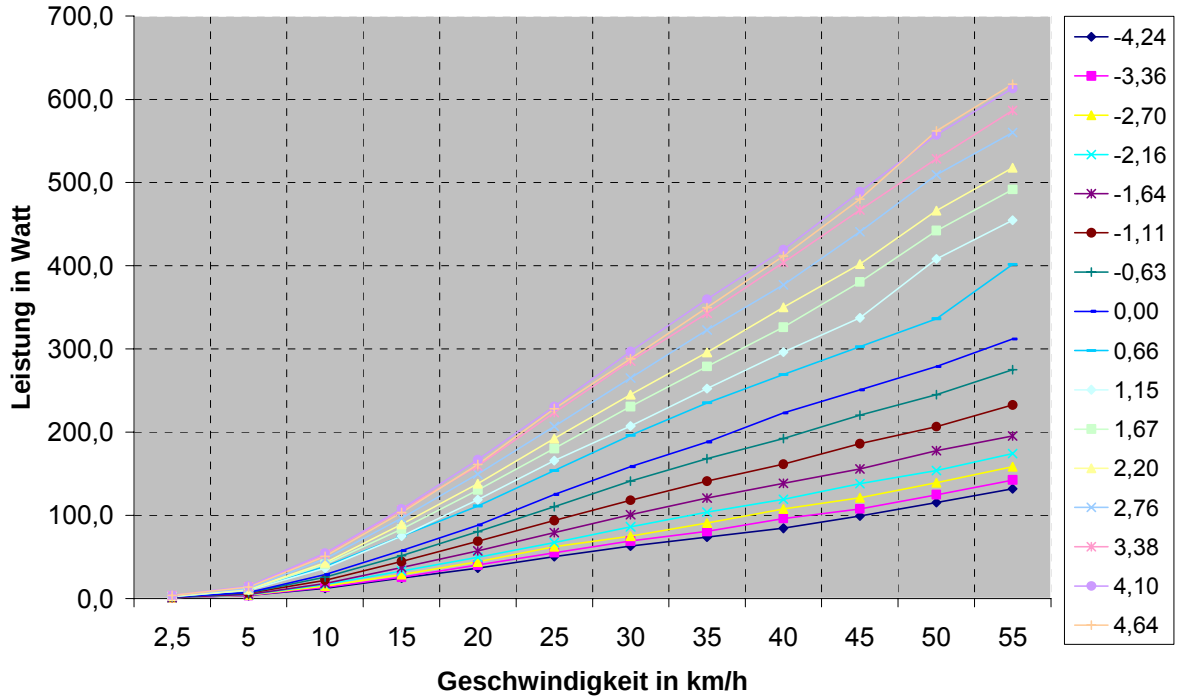


Abbildung 6.13: Kennfeld der Wirbelstrombremse (gemessen)

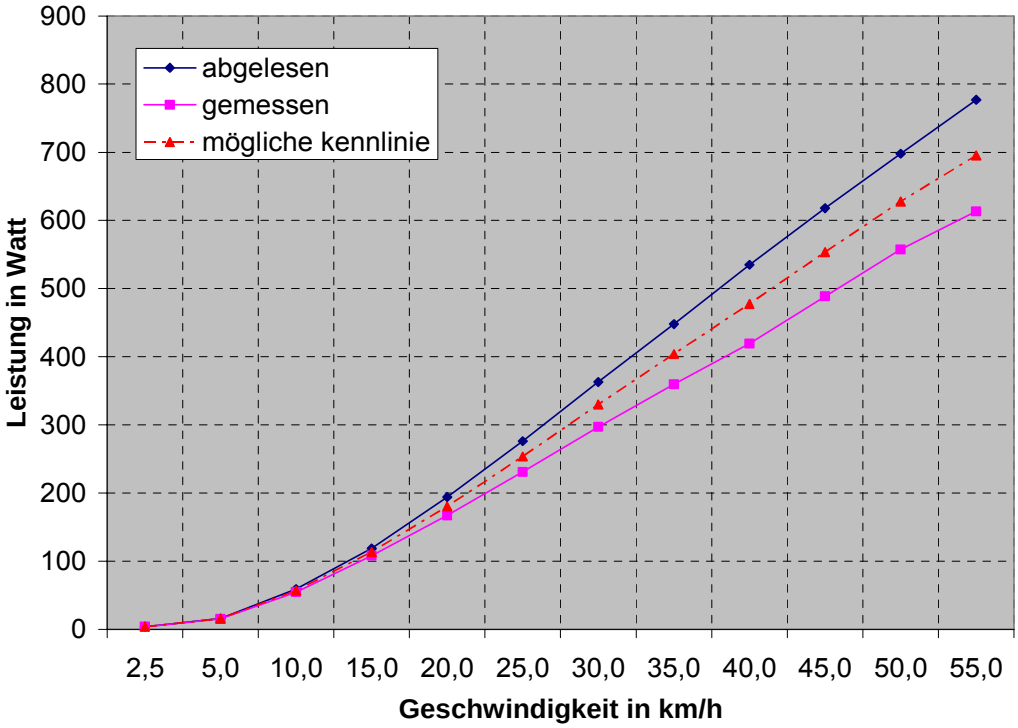


Abbildung 6.14: Kennlinien mit Pulsbreite von 4,10 ms

Um die beiden Ergebnisse direkt zu vergleichen, werden die Kennlinien von beiden Ergebnissen mit gleicher PWM-Pulsbreite von 4,10 ms in Abbildung 6.14 dargestellt. Aus der Abbildung ist zu erkennen, dass die Abweichung zwischen beiden Kennlinien mit zunehmender Geschwindigkeit wächst. Die Ergebnisse, die von dem Ergocomputer abgelesen wurden, scheinen zu groß zu sein. Die Ergebnisse, die gemessen wurden, sind möglicherweise zu klein. Die Gleichung (6.1) stellt die Leistungsbilanz in Fahrradtraining dar.

$$P_{\text{Mensch}} = P_{\text{Bremse}} + P'_{\text{Schlupf}} + P'_{\text{Kette}} \tag{6.1}$$

Hierbei ist P_{Mensch} die Leistung des Radfahrers, P_{Bremse} die Leistung der Wirbelstrombremse, P'_{Schlupf} die Verlustleistung des Schlupfs und P'_{Kette} die Verlustleistung der Kette. In dieser Leistungsbilanzgleichung, ist gemessene Leistung gleich der Leistung der Wirbelstrombremse; die Verlustleistungen von Schlupf und Kette sind noch unbekannt. Damit kann die Leistung des Radfahrers nicht genau bestimmt werden. Vermutlich liegt die Leistung des Radfahrers irgendwo zwischen den gemessenen und abgelesenen Ergebnissen, wie die rote Kennlinie in Abbildung 6.14 zeigt. Die Leistung des Radfahrers könnte bestimmt werden, wenn die Trittleistung oder das Trittmoment des Radfahrers gemessen werden können.

Um die Lookup-Table zu erstellen, werden in dieser Arbeit die Ergebnisse, die vom Ergocomputer abgelesen wurden, als Kennfeld der Wirbelstrombremse verwendet. Dies hat folgende Gründe:

1. Die gemessenen Ergebnisse sind fehlerhaft. Aus Abbildung 6.13 ist zu erkennen, dass die Kennlinien mit Pulsbreite von 0,66 ms und 1,15 ms teilweise nicht linear sind. Die Kennlinie mit einer Pulsbreite von 4,64 ms liegt teilweise unter der Kennlinie mit der Pulsbreite von 4,10 ms. Eigentlich sollte die Kennlinie mit größerer Pulsbreite oberhalb der Kennlinie kleiner Pulsbreite liegen.

2. Die gemessenen Ergebnisse sind nicht reproduzierbar. Wenn die Wirbelstrombremse zu unterschiedlichen Zeitpunkten mit gleicher Drehzahl und gleicher PWM-Pulsbreite betrieben wird, liefert sie unterschiedliche Leistungswerte.

3. Die Leistungskurven der Ergebnisse, die vom Ergocomputer abgelesen wurden, sind relativ linear und für unterschiedliche PWM-Pulsbreiten fast gleichmäßig verteilt.

Nach der Identifikation der Wirbelstrombremse, wird der Lookup-Table erstellt, um die Leistungssteuerung zu realisieren. Die Lookup-Table enthält nicht alle Werte der PWM-Pulsbreite und der Geschwindigkeit. Um beliebige Werte der PWM-Pulsbreite mit beliebiger Geschwindigkeit zu berechnen, wird das lineare Interpolationsverfahren verwendet. Die Erstellung des Lookup-Table und die lineare Interpolation zur Berechnung der PWM-Pulsbreite werden in einer MATLAB *s-Funktion* realisiert.

6.2.2.2 MATLAB *s-Funktion* in Form von level-1 *m-Files*

In dieser Arbeit wird die *s-Funktion* mit MATLAB in Form von level-1 *m-Files* geschrieben, da auf diese Weise keine Kompilierung gebraucht wird und alle Toolboxen, die von MATLAB angeboten werden, direkt in den Programmen verwendet werden können. Die *s-Funktion* in Form von level-1 *m-Files* hat folgende Standardform:

```
function [sys,x0,str,ts] = sfunname(t,x,u,flag,p1...)

switch flag,
    case 0, % Initialisierung und Zuweisung der Anfangszustände
        [sys,x0,str,ts] = [...];

    case 1, % Ableitungen der Zustandsgrößen
        sys = [...];

    case 2, % Aktualisierung der Zustandsgrößen
        sys = [...];

    case 3, % Rückgabe der Ausgangsgrößen
        sys = [...];

    case 4, % Rückgabe des nächsten Zeitpunkts für veränderliche Abtastzeit
        sys = [...];

    case 9, % Abbruch der s-Funktion
        sys = [...];

    otherwise, % unerwartete flags, Fehlerbehandlung
        error(['Unhandled flag = ',num2str(flag)]); % Fehlermeldungen
end;
```

Die im *m-File* vorkommenden Variablen haben folgende Bedeutungen:

sys : Rückgabeargument

*x*0 : Anfangszustand des Zustandvektors x

str : Reservierung für zukünftige Nutzung

ts : Matrix mit zwei Spalten, die die Abtastzeit und der Offset des Blocks enthält

t : Simulationszeit

x : Zustandsvektor

u : Eingangsvektor

flag : Integer dient zum Aufruf der einzelnen Teile (*case*) der *s-Funktion*

*p*1 : Benutzerparameter

Hier sind *t*, *x*, *u* und *flag* die Standardargumente, die eine *s-Funktion* immer beinhaltet. Das *flag* wird von der SIMULINK Simulationsumgebung automatisch gesetzt und dient zum Aufruf der einzelnen Teile der *s-Funktion*. In jedem Funktionsbereich (*case)*, der anhand der unterschiedlichen Werte von *flag* bestimmt wurde, werden die jeweils zurückgegebenen Vari-

ablen mit *sys* bezeichnet, d.h. z.B. im Abschnitt *case* 1 ist *sys* die Ableitung des Zustandsvektors *x*, im Abschnitt *case* 3 ist *sys* der Ausgangsvektor usw.

Die Initialisierung in *case* 0 wird nur zu Beginn der Simulation durchgeführt. Danach werden die Ableitungen der Zustandsgrößen in *case* 1, die Aktualisierung der Zustandsgrößen in *case* 2 und der Rückgabevorgang der Ausgangsgrößen in *case* 3 mit der in *ts* eingestellten Abtastzeit periodisch durchgeführt, bis die Simulationszeit abgelaufen ist. Der Benutzer kann während des Simulationsvorgangs durch Anklicken des Stoptasters die Simulation abbrechen. Dabei wird der Abbruchvorgang in *case* 9 durchgeführt.

Die Hauptaufgaben der *s-Funktion* dieser Masterarbeit sind der Datenaustausch zwischen Mikroprozessor und PC, die Darstellung der Trainingszustandmeldungen auf dem Display und die Implementierung der Leistungssteuerung. Es gibt in dieser *s-Funktion* keine kontinuierlichen oder diskreten Zustände. Das heißt, die Abschnitte *case* 1, *case* 2 und *case* 4 werden in dieser *s-Funktion* nicht auftreten.

Im Abschnitt *case 0* werden die folgenden Aufgaben abgearbeitet: die Initialisierung der seriellen UART Schnittstelle (*com* 2 des PC), die Erstellung des Lookup-Table und die Aktivierung der Benutzerschnittstelle. Die Initialisierung der UART Schnittstelle wird durch Verwendung der MATLAB Toolbox *serial* realisiert. Man braucht nur die geforderten Parameter für das Kommunikationsprotokoll in die Toolbox *serial* entsprechend zuzuweisen. Die Erstellung des Lookup-Table wird durch Verwendung der MATLAB Toolbox *interp*1 und *interp*2 realisiert. Die von der MATLAB Toolbox erstellte Benutzerschnittstelle wird durch Aufruf der Datei mit der Endung **.m* aktiviert. Nach Aktivierung der Benutzerschnittstelle bekommt die *s-Funktion* die sogenannten *handel* der GUI und aller verwendeten Schnittstellenelemente und kann damit in *case* 3 die entsprechenden Schnittstellenelemente in der GUI aktualisieren. Die Erstellung der Benutzerschnittstelle wird später im Abschnitt 6.2.2.3 *Erstellung der Benutzerschnittstelle(GUI)* noch ausführlich vorgestellt.

Die meisten Funktionen, wie der Datenaustausch mit dem Mikroprozessor, die Berechnung der Stellgrößen, die Leistungssteuerung, die Darstellung und die Aktualisierung der Trainingszustandsmeldungen und der Warnmeldungen, werden im Abschnitt *case* 3 realisiert. Der Programmablauf von Abschnitt *case* 3 dieser *s-Funktion* wird in Abbildung 6.15 dargestellt.

Dieses Programm wird mit einer in *ts* eingestellten Abtastzeit von 0,25 Sekunden periodisch durchgeführt. Zuerst werden die Trainingszeit und die Soll-Größen auf dem Display dargestellt. Dann wird abgefragt, ob sich im Empfangspuffer der *com* 2 Daten befinden. Ist dies nicht der Fall, so wartet das Programm solange bis Daten empfangen werden. Daraufhin werden diese Daten vom Empfangspuffer ausgelesen und der Puffer entleert. Die Werte der Trittfrequenz und der Herzfrequenz sowie der aktuelle Zustand des Geschwindigkeitszählers werden jeweils von den empfangenen Daten entnommen und in entsprechenden Variablen gespeichert. Danach wird die aktuelle Geschwindigkeit vom empfangenen Datum berechnet und anschließend alle Ist-Größen auf dem Display dargestellt. Daraufhin wird abgefragt, ob der Soll-Leistungswert zwischen 10 W und 1150 W liegt. Ist dies nicht der Fall, so wird weiter abgefragt, ob er kleiner als 10 W ist. Wenn dies der Fall ist, dann wird der PWM-Pulsbreite der minimale Wert zugewiesen und der Hinweis, dass der Soll-Leitungswert außerhalb des Betriebsbereichs liegt, ausgegeben. Wenn der Soll-Leitungswert nicht kleiner als 10 W ist, bedeutet dies, dass er größer als 1150 W ist. Dann wird der PWM-Pulsbreite der maximale Wert zugewiesen und der Hinweis, dass der Soll-Leitungswert außerhalb des Betriebsbereichs liegt, ausgegeben.

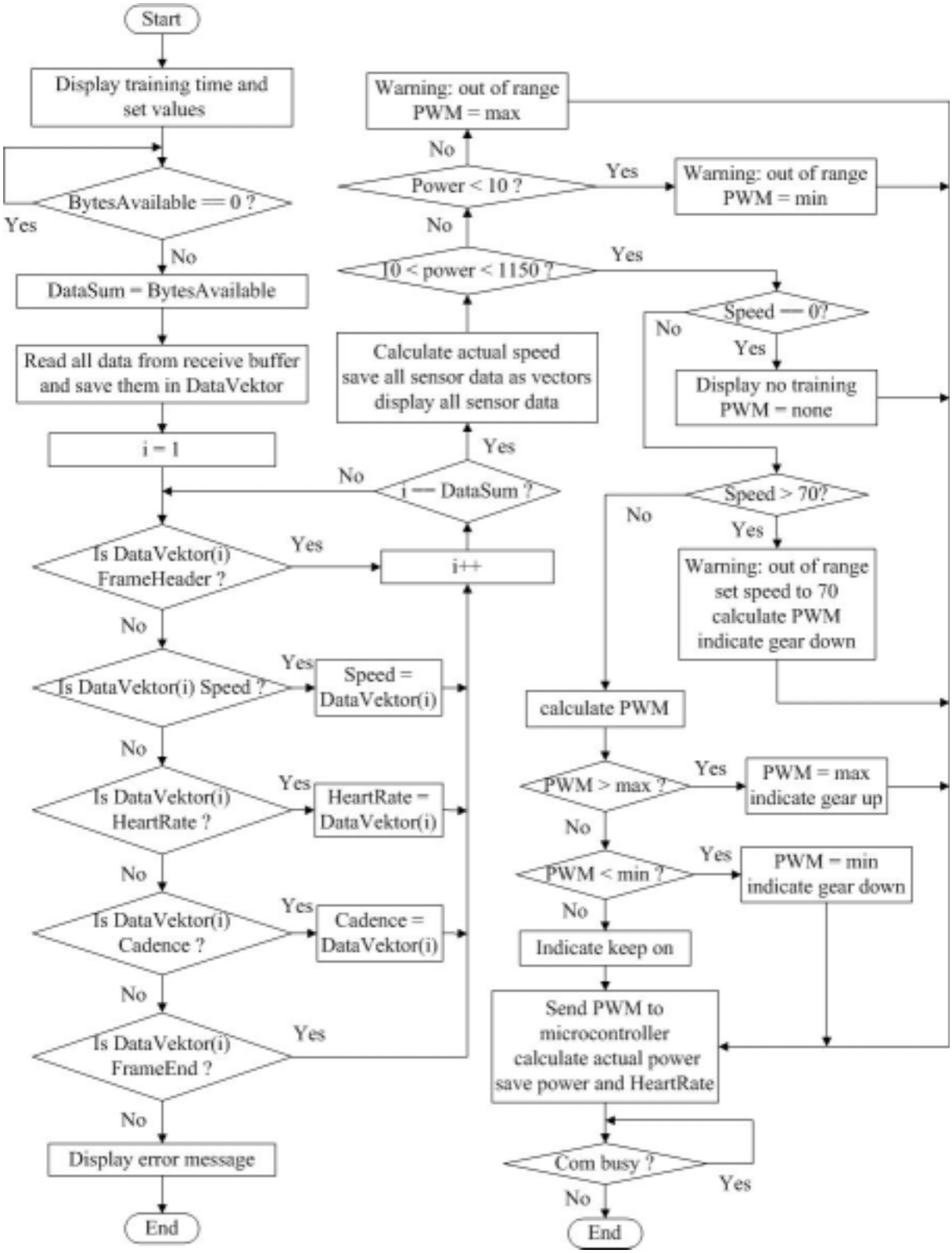


Abbildung 6.15: Programmablauf der *s-Funktion*

Wenn der Leistungswert zwischen 10 W und 1150 W liegt, so wird abgefragt, ob der Geschwindigkeitswert gleich null ist. Ist dies der Fall, so bedeutet dies, dass es kein Training

gibt. Dann muss kein PWM-Signal ausgegeben werden und der Text, dass kein Training stattfindet wird, als Hinweis ausgegeben. Wenn der Geschwindigkeitswert nicht gleich null ist, so wird weiter abgefragt, ob er größer als 70 km/h ist. Wenn dies der Fall ist, dann werden die Hinweise, dass die Geschwindigkeit außerhalb des Betriebsbereichs ist und die Übersetzung niedriger geschaltet werden sollte, ausgegeben. Weiterhin wird der Geschwindigkeitswert auf den maximalen Wert von 70 km/h eingestellt und die entsprechende PWM-Pulsbreite berechnet.

Wenn der Geschwindigkeitswert nicht größer als 70 km/h ist, dann wird die PWM-Pulsbreite durch lineare Interpolation des Lookup-Table berechnet. Danach wird abgefragt, ob die Pulsbreite größer als der maximale Wert ist. Ist dies der Fall, so wird die Pulsbreite auf den maximalen Wert eingestellt und die Aufforderung, langsamer zu treten, sowie der Hinweis, dass die Übersetzung höher geschaltet werden sollte, ausgegeben. Wenn die Pulsbreite kleiner als der maximale Wert ist, wird weiter abgefragt, ob sie kleiner als der minimale Wert ist. Ist dies der Fall, so wird die Pulsbreite auf den minimalen Wert eingestellt und die Aufforderung, schneller zu treten, sowie der Hinweis, die Übersetzung niedriger zu schalten, ausgegeben. Wenn die Pulsbreite größer als der minimale Wert ist, so bedeutet dies, dass die Wirbelstrombremse im zulässigen Bereich arbeitet. Dann werden die Hinweise, die Übersetzung und der Trainingsrhythmus beizubehalten, ausgegeben. Daraufhin wird das PWM-Signal an den Mikroprozessor gesendet und der Ist-Leistungswert durch lineare Interpolation des Lookup-Table berechnet. Anschließend werden alle Ist-Größen jeweils in einem Vektor gespeichert, um sie beim Ende des Trainings aufzuzeichnen. Das Programm wartet solange bis die *com* 2 frei ist und dann ist die Ausführung des Abschnittes *case* 3 dieser *s-Funktion* beendet.

Wenn die gesamte Simulationszeit abgelaufen ist oder der Benutzer während des Simulationsvorgangs den Stoptaster drückt, wird der Abbruchvorgang der *s-Funktion* in *case* 9 durchgeführt. Es wird zuerst der Hinweis, Training stoppt, ausgegeben. Dann werden die von *case* 3 gespeicherten Soll- und Ist-Größen in Diagramme aufgezeichnet und anschließend in Datei *simulat.mat* gespeichert. Zum Schluss werden alle verwendeten Variablen und *handel* der GUI gelöscht und die Speicherplätze freigegeben.

6.2.2.3 Erstellung der Benutzerschnittstelle (GUI)

Wie schon erwähnt, werden alle Soll- und Ist-Größen wie Leistung, Herzfrequenz, Geschwindigkeit und Trittfrequenz sowie Trainingszeit zahlenmäßig auf dem Display dargestellt. Weiterhin werden Warnmeldungen und Hinweise als Text ausgegeben. Außerdem wird die Bremsstärke als *Slider* mit den Stufen von minus sechs bis plus sieben angezeigt.

Zur Erstellung der Benutzerschnittstelle, wird in MATLAB die Toolbox GUIDE verwendet. Durch Eingabe des Befehls *guide* in das MATLAB *Command-Window* wird der GUIDE-Layout-Editor gestartet (siehe Abbildung 6.1). Danach werden die geforderten Schnittstellenelemente wie *Slider*, *Static Text*, *Panel* usw. von der Bibliothek der Schnittstellenelemente auf der linken Seite des Layout-Editors in das Editor-Fenster gezogen und dort angeordnet. Nach Abspeichern der erstellten GUI werden zwei Dateien mit eingegebenen Namen von GUIDE generiert, eine mit der Endung **.fig*, welche die Grafik mit der Anordnung der erstellten Elemente enthält und die andere mit der Endung **.m*, welche die sogenannten *Callbacks* dieser Elemente enthält.

In dieser Arbeit, werden nur die Schnittstellenelemente *Slider*, *Static Text* und *Panel* zur Erstellung der GUI verwendet. Die grafische Oberfläche der GUI ist in Abbildung 6.16 dargestellt. Die GUI kann in vier Zonen aufgeteilt werden: links ist die Zone für die Darstellung der Soll-Größen, in der Mitte ist die Zone für die Darstellung der Ist-Größen und der Hinweise zur Übersetzungsumschaltung des Fahrrades, rechts ist die Zone für die Darstellung der Wirbelstrombremse. Oben ist eine Zeile Text zur Darstellung der Warnungen oder Hinweise. Durch Verwendung der Schnittstellenelemente *Panel*, sind die ersten drei Zonen jeweils mit "Set Values", "Actual Values" und "Brake Status" bezeichnet. Die Wirbelstrombremse darf nur im Bereich von minus sechs bis plus sieben arbeiten. Dafür sind die oberen und unteren Grenzen in roter Farbe gekennzeichnet. Um den Radfahrer besser darauf aufmerken zu machen, werden alle Warnmeldungen ebenfalls in roter Farbe dargestellt.

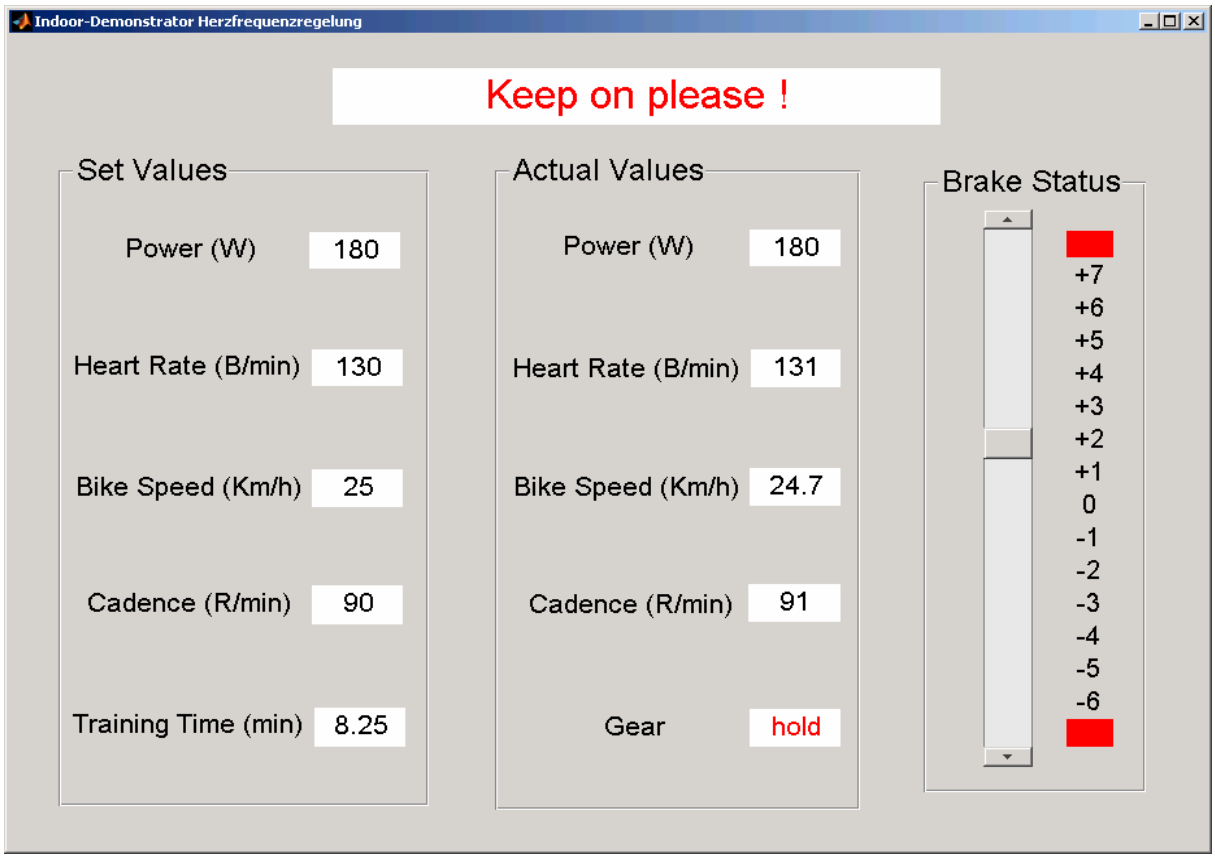


Abbildung 6.16: Benutzerschnittstelle

Nach Abspeichern der GUI, wird neben der Datei mit der Endung **.fig* noch eine Datei mit Endung **.m* generiert. Diese Datei wird in die *s-Funktion*, die in Abschnitt 6.2.2.2 vorgestellt wurde, aufgerufen und dadurch die *handel* der GUI und aller verwendeten Schnittstellenelemente zu dieser *s-Funktion* übergeben. Daraufhin kann alle Meldungen in die *s-Funktion* durch Verwendung der MATLAB Toolbox *set* dargestellt werden. In der Toolbox *set* können alle geforderten Eigenschaften der Schnittstellenelemente im entsprechenden Feld gesetzt werden. Zum Beispiel können die Texte der Hinweise mit Eigenschaft *string* in Schnittstellenelement *Static Text* ausgegeben werden. Die Schriftart, Schriftgrad, Schriftschnitt usw. von Texte können im bestimmten Eigenschaftsfeld der Toolbox *set* eingestellt werden. Auf eine genaue Beschreibung des Erstellungsvorgangs der Benutzerschnittstelle wird hier verzichtet und auf die sehr ausführliche Onlinehilfe der Software verwiesen.

6.3 Blockschaltbild des gesamten Regelungssystems

Bis jetzt wurden die in dieser Arbeit verwendeten Softwarekomponenten vorgestellt. Auf Basis dieser Software kann nun die Herzfrequenzregelung in SIMULINK implementiert werden. Das Blockschaltbild des gesamten Herzfrequenzregelkreises mit Anti-Windup-Maßnahme ist in Abbildung 6.17 dargestellt.

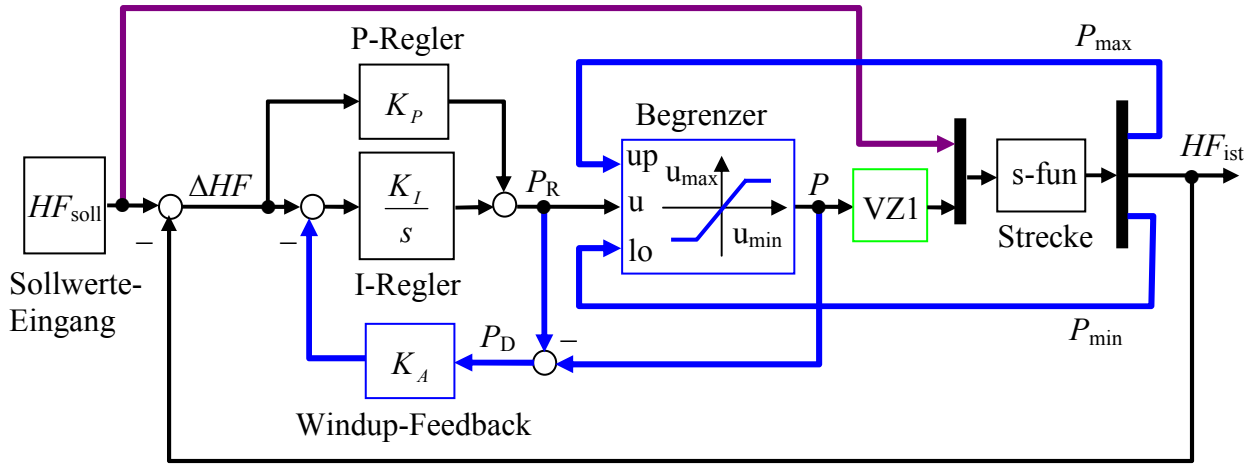


Abbildung 6.17: Das gesamte Regelungssystem mit Anti-Windup-Maßnahme

Die in schwarzer Farbe gezeichneten Pfeile und Blöcke bilden den grundlegenden Herzfrequenzregelkreis. Der PI-Regler wird in additiver Form dargestellt:

$$G_{PI}(s) = K_P + \frac{K_I}{s} \tag{6.2}$$

In den Block, der mit "Sollwerte-Eingang" benannt ist, können die Soll-Größen eingegeben werden. Die Soll-Herzfrequenz HF_{soll} wird mit der Ist-Herzfrequenz HF_{ist} verglichen, um die Regeldifferenz ΔHF zu bilden. Weiterhin werden alle Soll-Größen über den dunkel magentafarbenen Pfeil zur Darstellung auf dem Display an den Block "s-fun" geliefert.

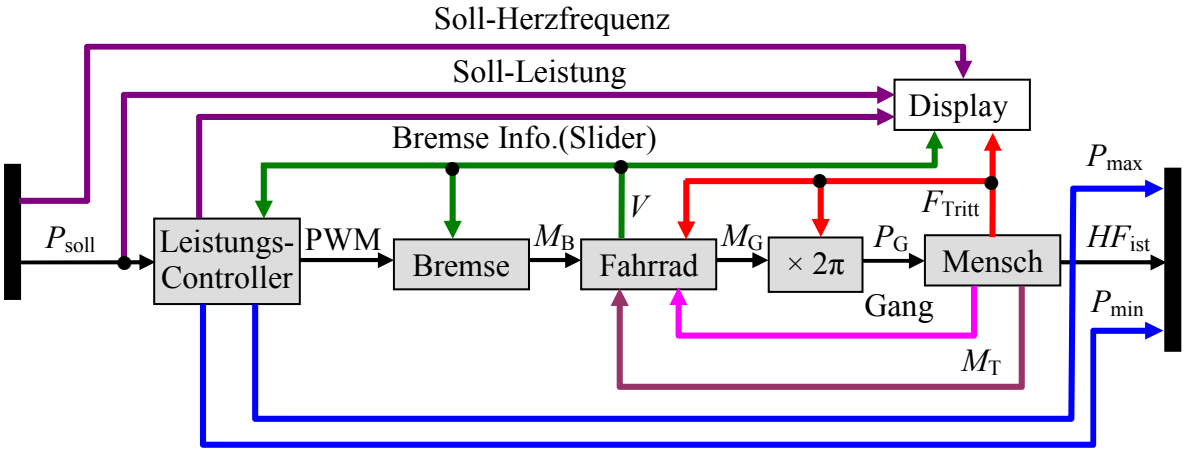


Abbildung 6.18: Blockbild der *s-Funktion*

Das Blockbild der *s-Funktion* ist in Abbildung 6.18 dargestellt. Hier werden die folgenden Aufgaben realisiert: Datenaustausch mit dem Mikroprozessor, Erstellung des Lookup-Table, Berechnungen der Stellgrößen und der Ist-Leistung durch lineare Interpolation, Leistungssteuerung, Darstellung und Aktualisierung der Trainingszustandsmeldungen. Außerdem werden die Grenzwerte des Sättigungsgliedes für die Anti-Windup-Maßnahme dynamisch festgestellt. Dies hat folgenden Grund:

- In dieser Arbeit, wird der Leistungswert als Stellgröße in dem Bereich von 10 W bis 1150 W begrenzt. Während des Trainings sollte der Radfahrer mit einer relativ konstanten Trittfrequenz fahren. Durch Übersetzung eines festen Gangs des Fahrrades ergibt sich eine relativ konstante Geschwindigkeit. Aus dieser konstanten Geschwindigkeit, wird die Leistung der Wirbelstrombremse auf den Bereich $P_{\min}$ bis $P_{\max}$ begrenzt. Das heißt, $P_{\min}$ und $P_{\max}$ geben den Bereich der Stellgröße vor. Während des Trainings, werden die Übersetzung des Fahrrades oder die Trittfrequenz des Radfahrers je nach unterschiedlichen Soll-Größen verändert. Dies führt zu einer Veränderung der Geschwindigkeit. Das heißt, dass $P_{\min}$ und $P_{\max}$ während des Trainings entsprechend der Geschwindigkeitsänderung angepasst werden. Die Grenzwerte des Sättigungsgliedes werden also dynamisch anhand der aktuellen Geschwindigkeit durch einen Lookup-Table festgelegt.

Die Anti-Windup-Maßnahme ist in die Abbildung 6.17 in blauer Farbe dargestellt. Hier sind P_{max} und P_{min} mit *up* und *lo* des Sättigungsgliedes verbunden, d.h. P_{max} und P_{min} sind die veränderlichen Grenzwerte des Sättigungsgliedes: $u_{\max} = P_{max}$ und $u_{\min} = P_{min}$. Wenn das Sättigungsglied nicht wirksam ist, ist der Reglerausgang P_R gleich dem Ausgang des Begrenzers P. Wenn das Sättigungsglied wirksam ist, ist P_R ungleich P. Durch Vergleich des Reglerausgangs P_R mit P ergibt sich die Differenz P_D. P_D wird mit einem Rückkoppelungsfaktor K_A multipliziert und an den Eingang des Integrators zurückgekoppelt. Der Faktor K_A ist so zu bestimmen, dass der Reglerausgang P_R am Grenzwert gehalten werden kann, wenn das Sättigungsglied wirksam wird.

6.4 Zusammenfassung

Hiermit ist die Beschreibung der Software für das Herzfrequenzregelungssystem des Indoor-Demonstrators abgeschlossen. Zuerst wurden die verwendete Entwicklungsumgebung und die Systemsoftwarekomponenten vorgestellt. Im Anschluss wurden die Programmabläufe der einzelnen Softwareteile von Mikroprozessor und MATLAB/SIMULINK erklärt. Zum Schluss wurde das gesamte Regelungssystem mit Anti-Windup-Maßnahme in SIMULINK vorgestellt. Die Beschreibung der Software ist nicht auf die eigentliche Programmierung eingegangen. Allerdings beinhaltet diese Beschreibung genügend Informationen, um die Entwicklung einer funktionsfähigen Software nachzuvollziehen. Für die Programmierung, soll nur noch darauf hingewiesen werden, dass es wichtig ist, möglichst wenige globale Variable in den C-Programmen für den Mikroprozessor zu verwenden. Für speziellere Fragen zum Programmablauf wird auf die für diese Arbeit erstellte CD-ROM mit den kompletten Programmen verwiesen. Im folgenden Kapitel wird die Inbetriebnahme der Herzfrequenzregelung für den Indoor-Demonstrator beschrieben.

7 Inbetriebnahme

In diesem Kapitel wird die Herzfrequenzregelung für den Indoor-Demonstrator in Betrieb genommen und das Validierungsergebnis präsentiert. Bis jetzt wurde die Erstellung und Programmierung des Herzfrequenzregelungssystems nur theoretisch beschrieben. Deshalb wird nun der praktische Teil der Inbetriebnahme vorgestellt.

An dieser Stelle wird das Verhalten des PI-Reglers gezeigt, welcher die Regelung der menschlichen Herzfrequenz übernimmt. Um die Regelgüte besser zu beurteilen, werden hierbei die Stellgröße, die Regelgröße, die Trittfrequenz und die Geschwindigkeit zusammen in einer Abbildung dargestellt und zwei Testergebnisse von einem bestimmten Testfahrer präsentiert.

Die zwei Tests wurden mit unterschiedlicher Soll-Herzfrequenz 35 Minuten lang an dem Indoor-Demonstrator-System durchgeführt. Bei beiden Tests, fuhr der Testfahrer mit einer relativ konstanten Trittfrequenz von 100 U/min. Bevor der Test gestartet wurde, hat sich der Testfahrer 10 Minuten warmgefahren. Seine Herzfrequenz betrug ungefähr 100 S/min.

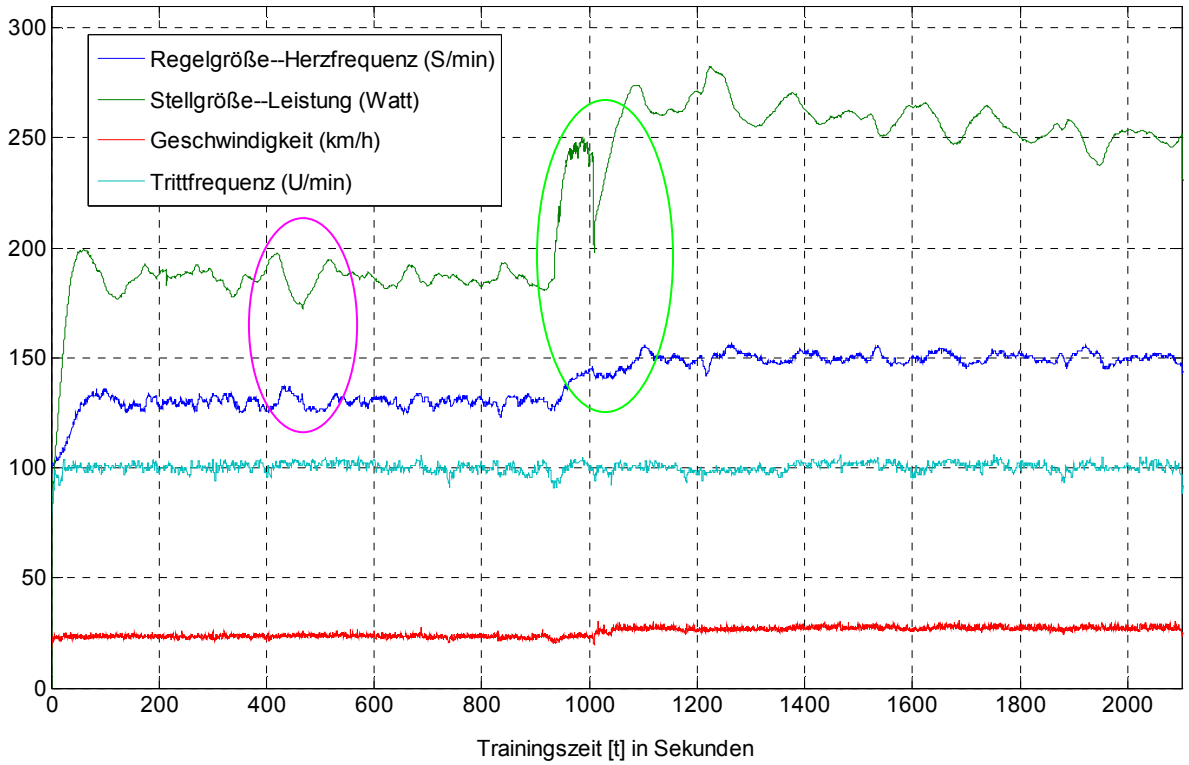


Abbildung 7.1: Verhalten des Reglers bei Erhöhung der Soll-Herzfrequenz

In Abbildung 7.1 sind der Verlauf der Herzfrequenz des Testfahrers und dessen Trittfrequenz und Leistung sowie die Geschwindigkeit des Rades des ersten Tests dargestellt. Bei diesem

Test wurde Soll-Herzfrequenz zweimal erhöht. Zu Beginn des Tests, wurde die Soll-Herzfrequenz von 100 S/min auf 130 S/min erhöht. Nach der ersten Testphase von 930 Sekunden, wurde die Soll-Herzfrequenz von 130 S/min auf 150 S/min erhöht. Durch die Erhöhung der Soll-Herzfrequenz von 100 S/min auf 130 S/min, steigt die Leistung mit einem größeren Überschwinger, um die Herzfrequenz des Fahrers auf Sollwert zu steigen. Es ist zu erkennen, dass die Herzfrequenz nie einen festen Wert erreicht, sondern immer leicht um den Sollwert schwingt. Während des Tests führen unvorhersehbaren Verhaltensweisen des Testfahrers, wie z.B. Schweißabwischen, Wassertrinken und Sitzpositionsveränderung, zur unerwarteten Schwankungen der Herzfrequenz. Dies ist in der Abbildung 7.1 mit Magenta markiert. Beim Zeitpunkt von ungefähr 420 Sekunden, ändert der Testfahrer seine Sitzposition. Dies führt zu einer großen Schwankung in seinem Herzfrequenzverlauf.

Durch die zweite Erhöhung der Soll-Herzfrequenz von 130 S/min auf 150 S/min zum Zeitpunkt von 930 Sekunden, steigt die Leistung an bis auf 250 Watt. Danach kann sie nicht mehr ansteigen. Weil bei diesem Leistungswert die Wirbelstrombremse an der oberen Grenze arbeitet und die Anti-Windup-Maßnahme wirksam wird. Nach der Umschaltung der Übersetzung des Fahrrades zum Zeitpunkt von 1000 Sekunden, steigt die Leistung weiter an und die Ist-Herzfrequenz folgt bis sie die Soll-Herzfrequenz erreicht. Dies ist der Abbildung in grün markiert. Außerdem kann man in der Abbildung erkennen, dass die Leistung des Fahrers bei konstanter Herzfrequenz mit zunehmender Testdauer sinkt.

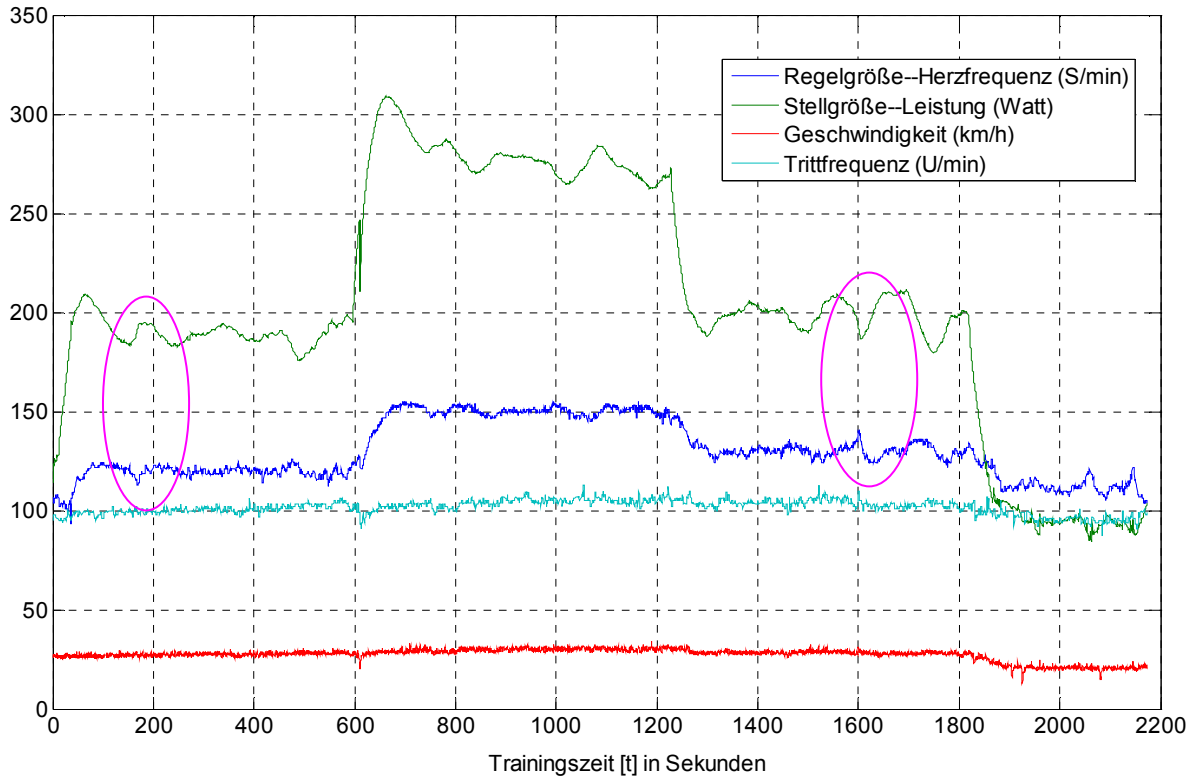


Abbildung 7.2: Verhalten des Reglers bei Erhöhung & Erniedrigung der Soll-Herzfrequenz

Die Ergebnisse des zweiten Tests sind in Abbildung 7.2 dargestellt. Der Unterschied zum ersten Test ist, dass dieser Test das Reglerverhalten bei Erniedrigung der Soll-Herzfrequenz auch anzeigt. Dieser Test wird in vier Testphasen aufgeteilt: die erste Testphase ist von 0 bis 600 Sekunden mit einer Soll-Herzfrequenz von 120 S/min, die zweite von 600 bis 1230 Se-

kunden mit einer Soll-Herzfrequenz von 150 S/min, die dritte von 1230 bis 1800 Sekunden mit einer Soll-Herzfrequenz von 130 S/min und die vierte von 1800 bis zum Ende mit einer Soll-Herzfrequenz von 110 S/min.

In der ersten und zweiten Testphase, erreicht die Ist-Herzfrequenz den Sollwert nach einem größeren Überschwinger der Stellgröße. Es ist an der Abbildung zu erkennen, dass die Überschwingweite der Stellgröße von der Sprungshöhe der Soll-Größe abhängt, je größer die Sprunghöhe der Soll-Größe ist, desto größer ist die Überschwingweite der Stellgröße. In der dritten und vierten Testphase, wird die Soll-Herzfrequenz um 20 S/min erniedrigt. Danach sinkt die Ist-Herzfrequenz entsprechend der Stellgröße. Die unerwartete Verhalten des Testfahrers sind auch in dieser Abbildung mit Magenta markiert, wobei die Ist-Herzfrequenz große Schwankungen hat. In Abbildung 7.2 ist noch deutlicher zu erkennen, dass die Leistung des Fahrers bei konstanter Herzfrequenz mit zunehmender Testdauer sinkt. Das bedeutet, dass mit zunehmender Testdauer, um eine konstante Leistung zu bringen, die Herzfrequenz des Testfahrers kontinuierlich steigt. Wie z.B. in der ersten und dritten Testphase, fährt der Fahrer mit ähnlich großer Leistung, aber die Herzfrequenz hat einen Unterschied von 10 S/min.

Die beiden Tests zeigen, dass der Herzfrequenzregler die Herzfrequenz des Radfahrers regeln kann, sowohl bei Erhöhung als auch bei Erniedrigung der Soll-Herzfrequenz. Hiermit ist die Inbetriebnahme der Herzfrequenzregelung für den Indoor-Demonstrator abgeschlossen. Die Ergebnisse der beiden Abbildungen zeigen, dass der Herzfrequenzregler die gestellte Aufgabe, die Herzfrequenz des Radfahrers auf den Sollwert einzuregeln, erfüllt. Im letzen Kapitel wird diese Masterarbeit zusammengefasst und einen Ausblick für weitere Entwicklungen gegeben

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In der vorliegenden Arbeit wurden die Konzeption und Realisierung einer Herzfrequenz-Regelung mit unterlagerter Leistungsregelung anhand eines Indoor-Demonstrators behandelt. Es wurde zuerst aus den Vorgaben der Aufgabenstellung ein Konzept erstellt, welches eine mögliche Realisierung dieses Problems darstellt. Auf der Basis des erarbeiteten Konzeptes wurden dann die vorgegebene Hardware an das System des Indoor-Demonstrators angeschlossen und die Sensordaten gemessen. Im Anschluss wurde das Kennfeld der Wirbelstrombremse identifiziert und die notwendige Software für die Leistungssteuerung erstellt. Danach wurden viele Stufentests mit unterschiedlichen Testprobanden durchgeführt und ein vernünftiges Menschmodell identifiziert. Daraufhin wurde ein PI-Regler als Herzfrequenz-Regler auf Basis der Leistungssteuerung in MATLAB/SIMULINK implementiert. Zum Abschluss wurde das gesamte Regelungssystem in Betrieb genommen und am Indoor-Demonstrator getestet. Mit dieser Arbeit steht ein System zur Verfügung, mit welchem es möglich ist die Herzfrequenz des Fahrers am Indoor-Demonstrator zu regeln. Dieses System kann auch als Basis für weitergehende Forschungen auf dem Gebiet des AmI-Szenarios *Assisted Training* dienen.

Das erarbeitete und umgesetzte Konzept hat sich bei der Inbetriebnahme als funktionsfähig erwiesen und erfüllt die Aufgabenstellung. Es haben sich während der Entwicklung einige Punkte gezeigt, die bei einer eventuellen Überarbeitung der Sensorplatine berücksichtigt werden sollten. Diese Punkte sind im Kapitel 4 *Konzeptentwurf des Regelungssystems* vermerkt.

8.2 Ausblick

Zum Abschluss dieser Arbeit wird noch ein Ausblick auf Themen gegeben, die in weiterführenden oder zukünftigen Arbeiten untersucht werden können. Durch die Verbesserung an der Sensorplatine können die Sensordaten genauer gemessen werden. Durch Verwendung von Filtern, können die Messfehler bei der Herzfrequenzmessung reduziert werden. Die Implementierung einer Leistungsregelung ist möglich, sobald die Trittleistung oder das Trittmoment des Radfahrers gemessen werden kann. Neben der Hardwareverbesserung, kann auch einiges an der Software untersucht werden. Es können zum Beispiel andere Regelalgorithmen, wie der Modell Prädiktiv Controller, für die Herzfrequenzregelung verwendet werden. Ein weiterer Ansatz für weitere Entwickelungen kann die Online-Mensch-Modellierung und Reglerparameter-Anpassung sein. Damit kann jeder individuelle Radfahrer durch einen Stufentest vor dem Training sein eigenes Körpermodell erstellen, und dann durch Online-Anpassung eigene für ihn zutreffende Reglerparameter zu erhalten. Als Basis für weitere Arbeiten im Rahmen des AmI-Szenarios *Assisted Training* kann das Regelungssystem auf MICA-Knoten implementiert werden.

Tabellen- und Abbildungsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Atm-01] Atmel: AVR ISP User Guide. Rev. 2468A-09/01/2M, San Jose: Atmel Corperation, 2001.

[Atm-03] Atmel: ATmega 128, ATmega128L (Datenblatt). Rev. 2467H-AVR-02/03, San Jose: Atmel Corperation, 2003.

[Föl-85] Föllinger, Otto: Regelungstechnik: Einführung in die Methoden und ihre Anwendung. 5. Auflage, Dr. Hüthig Verlag Heidelberg, 1985, ISBN 3-7785-1137-8.

[Föl-93] Föllinger, Otto: Lineare Abtastsysteme. R. Oldenbourg Verlag GmbH, München, 1993, ISBN 3-486-22725-4.

[Ima-00] ImageCraft: ImageCraft AVR C compiler and development environment for Atmel AVR. Version 6, ImageCraft Creations Inc.,2000.

[Lin-00] Lindner, Wolfram: Radsporttraining. BLV Verlagsgesellschaft mbH, München 2000, ISBN 3-405-15836-2.

[Lit-05] Litz, Lothar: Grundlagen der Automatisierungstechnik. R. Oldenbourg Verlag, 2005, ISBN 3-486-27383-3.

[Web-01] http://www.atmel.com/products/avr/,
> Tools & Software > AVR Studio 4
Zugriff am 03.06.2005.

[Web-02] http://www.medion.de/,
> Service&Support > Treiber&Updates > Volltextsuche: md 3680 > Download
Zugriff am 01.06.2005.

[Web-03] http://www.tacx.nl,
Zugriff am: 01.06.2005.

[Web-04] http://www.eit.uni-kl.de/AmI/, Homepage des Forschungsbereichs AmI.
Zugriff am: 04.06.2005.

[Web-05] http://www.mathworks.com/access/helpdesk/help
> MATLAB
> MATLAB Toolboxes
> Simulink
Zugriff am 21.06.2005

Kontakt

Prof. Dr.-Ing. habil. Lothar Litz
Technische Universität Kaiserslautern
Lehrstuhl für Automatisierungstechnik
Postfach 3049
67653 Kaiserslautern
Tel.: +49 (0) 631/205-4450
Fax.: +49 (0) 631/205-4462
E-Mail: litz@eit.uni-kl.de
URL: http://www.eit.uni-kl.de/litz/

Dipl.-Ing. Oliver Gabel
Adresse wie oben
Tel.: +49 (0) 631/205-4459
Fax.: +49 (0) 631/205-4462
E-Mail: gabel@eit.uni-kl.de
URL: http://www.eit.uni-kl.de/litz/